SECTION 2
SYSTEM OVERVIEW

2.1 SYSTEM APPLICATION. The DSRS is the product of an evolution of software developed under the DISA/JIEO/CFSW Software Reuse Program. It is an automated repository designed to support a strategy of asset reuse. Reuse is the application of previously developed assets to a new system or an expansion and/or enhancement of an existing system. A software reuse program may take the following approach to implement this strategy:

a. Identify reuse opportunities through avenues such as domain analysis;

b. Provide users with a facility to find, evaluate, and acquire reusable assets;

c. Provide asset configuration management and quality assurance;

d. Establish policies and procedures; and

e. Provide reuse training and support.

The DSRS supports this strategy by providing a facility for finding, evaluating, and storing reusable assets. Additionally, the DSRS supports asset configuration management.

The DSRS supports the distribution of reusable assets through two automated tools:

a. The DSRS Librarian tool is a mechanism of the software reuse program which provides for:

(1) Managing the catalog of assets, and

(2) Reports on usage of the system.

b. The DSRS User tool is outside the scope of the software reuse program and provides for:

(1) Users to request and obtain assets.

The overall objective of the SRP is to promote the application of reuse to reduce the costs associated with system development. The DSRS will automate access to reusable assets. The DSRS will perform the following functions:

a. Provide users with a repository system in which multiple users can simultaneously search for, evaluate, and acquire reusable assets;

b. Provide a tool to perform various maintenance activities for the repository of reusable assets; and

c. Provide the capability to interact with other DSRS sites and other interoperable repositories to increase the availability of reusable assets.

The system will provide a repository in which multiple users can simultaneously search for, evaluate, and acquire reusable assets and in which the DSRS support staff can perform various maintenance activities for the repository of assets.

The DSRS will provide a structured querying mechanism with a user-friendly interface. Figure 2-1 displays the basic DSRS architecture, which lists each executable and the mapping of each executable to the user interface, DSRS software, Database Management System (DBMS), and the operating system.

The User Tool will be implemented on a Sun/UNIX workstation to support a Motif-based interface; and implemented for a MS-Windows workstation to support a Windows-based interface. The User Tool functionality will be consistent between each interface supported, unless limited by the host environment.

The Librarian Tool will be implemented for a MS-Windows workstation to support a Windows-based interface.

The DSRS Server will be built on a relational database, which organizes information about the reusable assets. The reusable asset files (including the actual asset, an abstract file, and document files) will be stored in the underlying file system provided by the operating system software and will be transparent to the user.

2.2 SYSTEM ORGANIZATION. Data processing operations for the DSRS are organized and invoked by different executables. These executables fulfill the major functions provided by the system. Figure 2-1 displays the basic DSRS architecture depicting these executables. The purpose of each executable and the major functions implemented by each executable are addressed in the following subsections.

DSRS Applications	User		Librarian
Client Executables	DSRS	DSRS	DSRSLIB
Client HW/OS*	PC/DOS	Sun/UNIX	PC/DOS
User Interface	MS-Windows/ Windows NT	X/Motif	MS-Windows/ Windows NT
DBMS	Oracle/Access	Oracle/MSQL	Oracle
Server Executables	pc_server or pc_server.sol23 and dsrs_server or dsrs_server.sol23		lib_server or lib_server.sol23
Server HW/OS*	Sun/UNIX		

* Refer to Section 5.2 of the *Functional Description for the DSRS* for environment specifics.

Figure 2-1. Basic DSRS Architecture

2.2.1 User Interaction.

a. Search Mechanisms
   (1) List RA IDs
   (2) List RA Names
   (3) Find Keywords
   (4) Catalog Indexes (WWW)
b. Browse information about an RA
c. Tools
   (1) Analyze retrieved RAs based on available metrics
   (2) Clear local database
   (3) Sort Candidate List by RA ID or RA Name
d. Extract local, remote, or foreign RAs. The extraction of remote and foreign RAs will be managed and controlled by the TCP/IP Asset Transfer Interface
e. Provide storage and retrieval of session information
f. Provide options for preferences (defaults), network, and WWW configurations
g. Provide on-line help and a tutorial

2.2.2 Librarian Interaction.

a. RAs
   (1) RA Information
   (2) RA to Domain Assignments
   (3) Related RAs
   (4) RA Metrics
   (5) RA Files
b. Group Information and Group Assignments
c. User Information and User Assignments
d. Site Information and Site Assignments
e. Domains
f. Metrics
g. Relationships
h. Logs
i. Provide Import of RAs
j. Create Export files
k. Reports
l. Provide option to set server configuration
m. Provide on-line help and a tutorial

2.2.3 Server Interaction.

a. Authentication Requests
b. Catalog Requests
c. File Transfer Requests

2.3 SYSTEM FUNCTION. This section describes the Top-Level Software Units (TLSUs) and the interrelationships among those TLSUs used in the implementation of the DSRS. According to the Booch method, a TLSU is represented by a class.

2.3.1 Classes (TLSUs). A class captures the common structure and common behavior of a set of objects. It is an abstraction of real-world items. Each class encapsulates the subset of operations necessary to implement the system.

2.3.1.1 User Tool. The *User Tool* is implemented in two platforms: MS-Windows and X/Motif. However, there are three executables produced: User Tool for MS-Windows, User Tool for Unix SunOS, and User Tool for Unix Solaris. The Unix SunOS version and Unix Solaris version are identical in their design and implementation. Therefore, those two versions are covered under the heading: X/Motif. The following paragraphs describe the classes under their appropriate headings.

2.3.1.1.1 MS-Windows. The top-level classes for the DSRS User for MS-Windows are as follows:

2.3.1.1.1.1 DSRSUserApp. *DSRSUserApp* is a container class. It functions as a folder containing all classes and other files related to the application. There are no operations associated with it.

2.3.1.1.1.2 FrmBrMet. *FrmBrMet* is a class representing the *Browse Metrics* screen. Its function is to enable the user to browse the metric(s) of an RA.

2.3.1.1.1.3 FrmWldName. *FrmWldName* is a class representing the *List RA Names* screen. Its function is to allow the user to search for RAs by RA Name and list the results.

2.3.1.1.1.4 FrmWldID. *FrmWldID* is a class representing the *List RA IDs* screen. Its function is to allow the user to search for RAs by RA ID and list the results.

2.3.1.1.1.5 FrmNetDialog. *FrmNetDialog* is a class representing the *Network Configuration* screen. Its function is to allow the user to change the default network configuration.

2.3.1.1.1.6 FrmLogin. *FrmLogin* is a class representing the *Login* screen. Its function is to enable the user to connect to a server.

2.3.1.1.1.7 FrmAbout. *FrmAbout* is a class representing the *About* screen. It contains information about the application, such as version number and release date.

2.3.1.1.1.8 FrmExtract. *FrmExtract* is a class representing the *Extract RAs* screen. Its function is to allow the user to select multiple RAs from the candidate list to be extracted as described in Figure 2-3.

2.3.1.1.1.9 FrmKeywordD. *FrmKeywordD* is a class representing the *Keyword Search* screen. Its function is to allow the user to search for RAs based on keywords entered, and list the results.

2.3.1.1.1.10 FrmBrDoc. *FrmBrDoc* is a class representing the *Browse File List* screen. Its function is to enable the user to browse the list of files available for an RA and view the content.

2.3.1.1.1.11 FrmBrRel. *FrmBrRel* is a class representing the *Browse Related RA* screen. Its function is to enable the user to browse the list of related RA(s) and the kind of relationship(s) for a selected RA.

2.3.1.1.1.12 FrmPreferences. *FrmPreferences* is a class representing the *Preferences* screen. Its function is to allow the user to change the default domain, search threshold notification, and default text viewer.

2.3.1.1.1.13 FrmWWWCa. *FrmWWWCa* is a class representing the *World Wide Web Catalog* screen. Its function is to enable the user to select the World Wide Web browser and determine the default home page.

2.3.1.1.1.14 FrmChgPa. *FrmChgPa* is a class representing the *Change Password* screen. Its function is to allow the user to change the password.

2.3.1.1.1.15 FrmAnalyzed. *FrmAnalyzed* is a class representing the *Analyze* screen. Its function is to allow the user to analyze retrieved RAs based on available metrics and sort them according to the available options.

2.3.1.1.1.16 FrmBrAbst. *FrmBrAbst* is a class representing the *Abstract* screen. Its function is to allow the user to view the abstract of an RA.

2.3.1.1.1.17 FrmUserMain. *FrmUserMain* is a class representing the main screen of the application. Its function is to allow the user to access the functionalities of the *User Tool* described in Figure 2-2 through menu items of the menubar or command buttons on the toolbar.

2.3.1.1.2 X/Motif. The top-level classes for the DSRS User for X/Motif are as follows:

2.3.1.1.2.1 UDSRSApp. *UDSRSApp* is a class representing the application-level class, which is the highest in the application framework hierarchy. It is an abstract class and cannot be instantiated. Each application can have one application object only. This class handles the start-up and shut-down activities of an application.

2.3.1.1.2.2 USession. *USession* is a class representing the document-level class. It resides at the second layer of the application framework hierarchy. This class functions as a central means of communication for changes and updates in different window-level classes.

2.3.1.1.2.3 USessWin. *USessWin* is a window-level class representing the main screen of the application. Its function is to allow the user to access the functionalities of the *User Tool* described in Figure 2-2 through menu items of the menubar or command buttons on the toolbar.

2.3.1.1.2.4 UDSRSWin. *UDSRSWin* is a window-level class representing the *About* screen. It contains information about the application, such as version number and release date.

2.3.1.1.2.5 UWidWin. *UWidWin* is a window-level class representing the *List RA IDs* screen. Its function is to allow the user to search for RAs by RA Names and list the results.

2.3.1.1.2.6 UPrefWin. *UPrefWin* is a window-level class representing the *Preferences* screen. Its function is to allow the user to change the default domain, search threshold notification, and default text viewer.

2.3.1.1.2.7 UCngPWin. *UCngPWin* is a window-level class representing the *Change Password* screen. Its function is to allow the user to change the password.

2.3.1.1.2.8 UWWWWin. *UWWWWin* is a window-level class representing the *World Wide Web Catalog* screen. Its function is to enable the user to select the World Wide Web browser and determine the default home page.

2.3.1.1.2.9 URelWin. *URelWin* is a window-level class representing the *Browse Related RA* screen. Its function is to enable the user to browse the list of related RA(s) and the kind of relationship(s) for a selected RA.

2.3.1.1.2.10 UMetricWin. *UMetricWin* is a window-level class representing the *Browse Metrics* screen. Its function is to enable the user to browse the metric(s) of an RA.

2.3.1.1.2.11 UAbstWin. *UAbstWin* is a window-level class representing the *Abstract* screen. Its function is to allow the user to view the abstract of an RA.

2.3.1.1.2.12 GNetWin. *GNetWin* is a window-level class representing the *Analyze* screen. Its function is to allow the user to analyze retrieved RAs based on available metrics and sort them according to the available options.

2.3.1.1.2.13 UXtractWin. *UXtractWin* is a window-level class representing the *Extract RAs* screen. Its function is to allow the user to select multiple RAs from the candidate list to be extracted as described in Figure 2-3.

2.3.1.1.2.14 UFileWin. *UFileWin* is a window-level class representing the *Browse File List* screen. Its function is to enable the user to browse the list of files available for an RA and view the content.

2.3.1.1.2.15 UFindWin. *UFindWin* is a window-level class representing the *Keyword Search* screen. Its function is to allow the user to search for RAs based on keywords entered, and list the results.

2.3.1.1.2.16 UNamesWin. *UNamesWin* is a window-level class representing the *List RA Names* screen. Its function is to allow the user to search for RAs by RA ID and list the results.

2.3.1.1.2.17 ULogWin. *ULogWin* is a window-level class representing the *Login* screen. Its function is to enable the user to be connected to a server.

2.3.1.1.2.18 UNetWin. *UNetWin* is a window-level class representing the *Network Configuration* screen. Its function is to allow the user to change the default network configuration.

2.3.1.1.2.19 UServer. *UServer* is a control class which handles the communication between the DSRS User for X/Motif and the server code.

2.3.1.1.2.20 ULocalDB. *ULocalDB* is a control class which handles the communication between the DSRS User for X/Motif and the local database (MSQL).

2.3.1.2 Librarian Tool. The *Librarian Tool* is implemented in one platform only: MS-Windows. The following paragraph lists and describes top-level classes developed for the DSRS Librarian for MS-Windows.

2.3.1.2.1 DSRSLibApp. *DSRSLibApp* is a container class. It functions as a folder containing all classes and other files related to the application. There are no operations associated with it.

2.3.1.2.2 FrmLibMain. *FrmLibMain* is a class representing the main screen of the application. Its function is to allow the Librarian/Supervisor to access the functionalities of the *Librarian Tool* described in Figure 2-5 through menu items of the menubar or command buttons on the toolbar.

2.3.1.2.3 FrmLogin. *FrmLogin* is a class representing the *Login* screen of the application. Its function is to enable the Librarian/Supervisor to connect to a server and a database.

2.3.1.2.4 FrmRA. *FrmRA* is a class representing the *RAs* screen of the application. Its function is to enable the Librarian/Supervisor to maintain the RAs and RA Information in the local catalog.

2.3.1.2.5 FrmGroup. *FrmGroup* is a class representing the *Groups* screen of the application. Its function is to enable the Librarian/Supervisor to maintain group information and group assignments for RAs, users, and sites.

2.3.1.2.6 FrmUsers. *FrmUsers* is a class representing the *Users* screen of the application. Its function is to enable the Librarian/Supervisor to maintain user information and group assignments for local users.

2.3.1.2.7 FrmSites. *FrmSites* is a class representing the *Sites* screen of the application. Its function is to enable the Librarian/Supervisor to maintain the information for local, remote DSRS sites, and foreign sites (other interoperable repositories).

2.3.1.2.8 FrmNewDomain. *FrmNewDomain* is a class representing the *Domains* screen of the application. Its function is to enable the Librarian/Supervisor to maintain the domains.

2.3.1.2.9 FrmMetrics. *FrmMetrics* is a class representing the *Metrics* screen of the application. Its function is to enable the Librarian/Supervisor to maintain the types of metrics for the local site.

2.3.1.2.10 FrmRelation. *FrmRelation* is a class representing the *Relationships* screen of the application. Its function is to enable the Librarian/Supervisor to maintain the types of relationships which can be used when entering related RAs.

2.3.1.2.11 FrmImport. *FrmImport* is a class representing the *Import* screen of the application. Its function is to enable the Librarian/Supervisor to read RA information contained in an export file (created at another DSRS) and write the information to the local catalog.

2.3.1.2.12 FrmExport. *FrmExport* is a class representing the *Export* screen of the application. Its function is to enable the Librarian/Supervisor to submit assets to other DSRS sites via the Export function. The export function will create an export file containing RAs and associated information that will be copied (imported) into another DSRS catalog. The Export function will also create indexed files that will be copied (imported) into other interoperable repositories (foreign sites).

2.3.1.2.13 FrmUsage. *FrmUsage* is a class representing the *Usage* screen of the application. It functions as a log of RA extractions.

2.3.1.2.14 FrmAbout. *FrmAbout* is a class representing the *About* screen of the application. It contains information about the application, such as version number and release date.

2.3.2 DSRS Class Diagram with Relationships. Classes do not exist in isolation. Rather they are related in a variety of ways to form the class structure of the domain. Relationships help further define the classes by exposing their content or dependency on the content of others. Figure 2-6, Figure 2-7, and Figure 2-8 illustrate class diagrams with relationships. These class diagrams are the actual diagrams used in implementation. They look different from the class diagrams produced during the analysis stage. Refer to Section 4 of the *System Specification for the DSRS* for detailed information about the analysis stage.

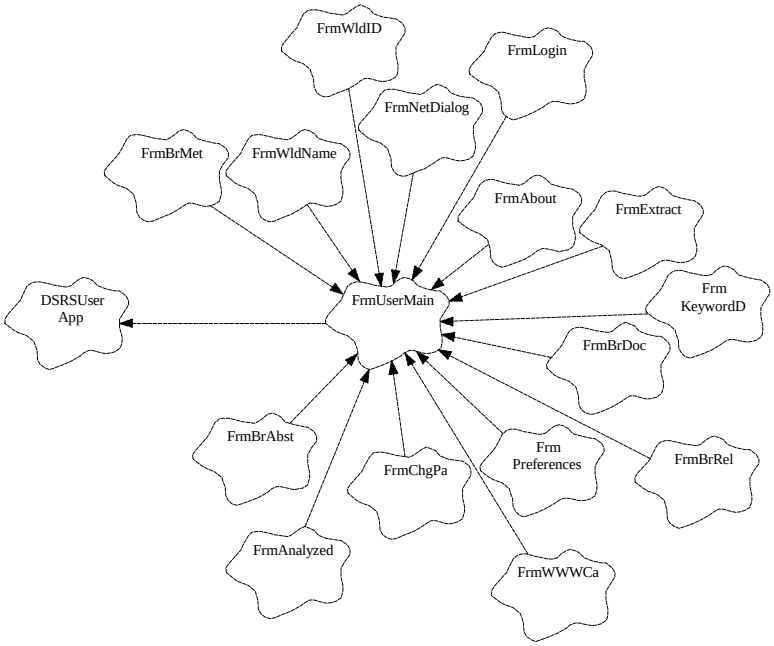


Figure 2-2. DSRS User for MS-Windows Class Diagram with Relationships

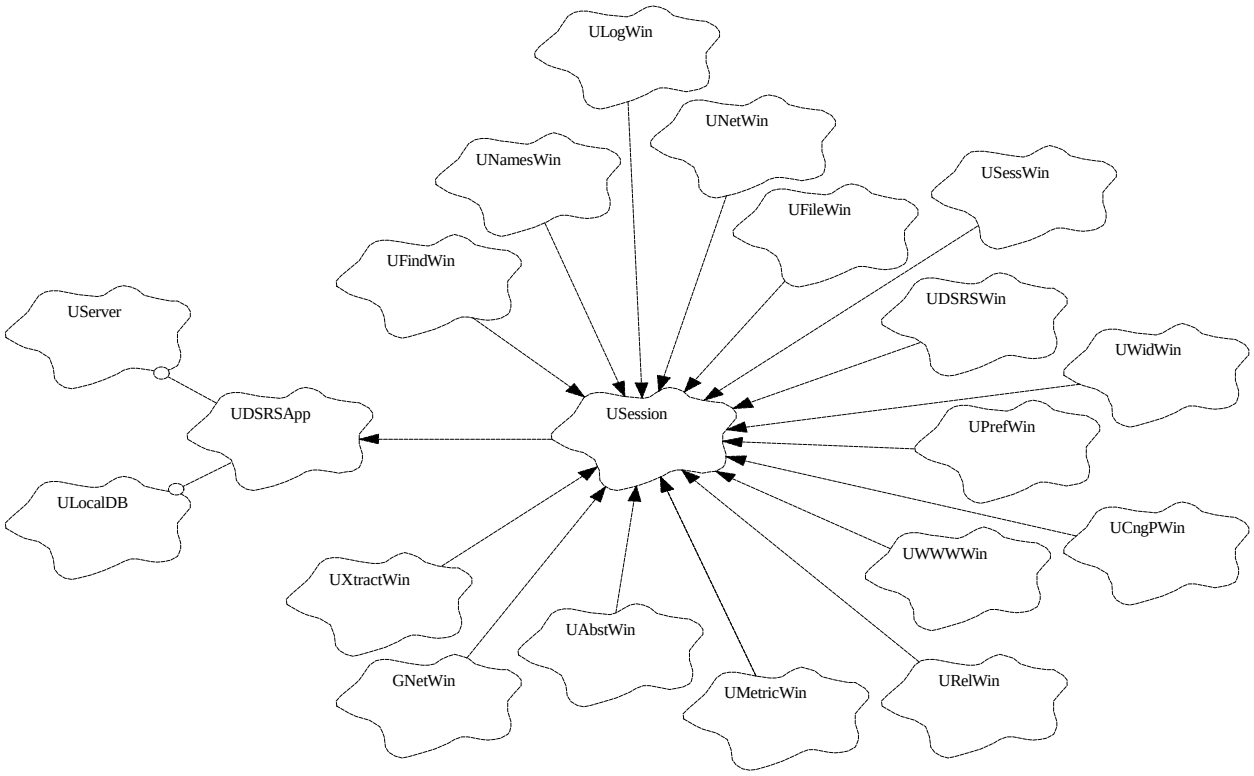


Figure 2-3. DSRS User for X-Windows/Motif Class Diagram with Relationships

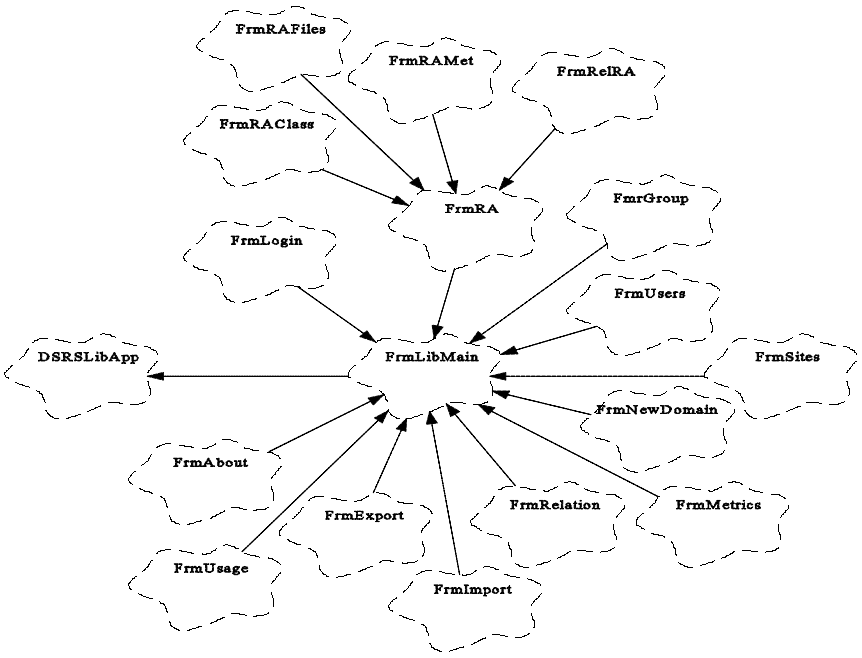


Figure 2-4. DSRS Librarian for MS-Windows Class Diagram with Relationships

2.4 INFORMATION INVENTORY.

2.4.1 Database Management System.

2.4.1.1 Database Files. ORACLE RDBMS uses several features of the SunOS and Solaris operating systems to provide a secure environment for the user community. The SunOS and Solaris features include file ownership, group accounts, and the ability to have a program change its user-id upon execution. The directories containing the database files must be owned by the *oracle* user-id. Group and world users should not have write access to the database. Table 2-I indicates all permanent database files which are used by the system.

a. The **Database** row identifies the database instance used by DSRS.

b. The **Tablespace** row identifies the names of the four tablespaces contained in the DSRS database. The SYSTEM tablespace exists in every ORACLE database and is created automatically during installation. The SYSTEM tablespace contains the data dictionary for the entire DSRS database, with names and locations of all database objects (such as tables, indexes, and other tablespaces). The other three tablespaces are unique to the DSRS.

c. The **Files** row identifies the name of the database file.

d. The **Directory** row identifies the directory structure where the database files are located. The logicals $DB_LOC and $INDEX_LOC are defined in the script **/dsrscm/testdb/dsrs_install**. Refer to the *Implementation Procedures for the DSRS and Related COTS Support Software* for additional information.

e. The **Size** row identifies the size of the database file in megabytes.

Table 2-I. ORACLE Database Structure

Database	**DSRS**			
Tablespaces	SYSTEM	DSRS	DSRSINDXS	DSRS_ROLLBK
Files	dbf [ORACLE_SID]* .dbf	[ORACLE_SID]* .dbf	[ORACLE_SID]* indxs.dbf	[ORACLE_SID]* _rollbk.dbf
Directory		$DB_LOC	$INDEX_LOC	$DB_LOC
Size	40M	60M	60M	60M

*[ORACLE_SID] is replaced with the value that the ORACLE_SID variable is set to.

2.4.1.2 ORACLE Database Startup for SunOS 4.1.3. ORACLE provides a shell script to ensure a clean, automatic database startup. To ensure that the ORACLE database will come up after the SunOS has been shutdown and restarted, the **dbstart** shell script can be added to the **/etc/rc.local** file. This script will bring up all of the databases listed in **/etc/oratab** file that have the third field specified as **Y** (yes).

The **dbstart** script can also be executed when the database is down or has been shutdown with the **dbshut** command described below.

The entries in the **/etc/oratab** file have the syntax:

$ORACLE_SID:$ORACLE_HOME:<N|Y>

The following line interactively starts up the databases when executed by the superuser or inserted into the **/etc/rc.local** script file:

su - oracle -c /usr/oracle/bin/dbstart

The ORACLE database can also be started within the SQL*DBA tool. The SQL*DBA is a tool for helping database administrators manage and monitor a database. SQL*DBA may be used interactively or an SQLDBA command line may be executed at the operating system level. Refer to the *ORACLE RDBMS Database Administrator's Guide* for detailed information on the SQL*DBA tool.

2.4.1.3 ORACLE Database Startup for Solaris 2.3. ORACLE provides a shell script to ensure a clean, automatic database startup. To ensure that the ORACLE database will come up after the Solaris has been shutdown and restarted, the dbstart shell script can be linked to files in **/etc/rc2.d** file. This script will bring up all the databases listed in **/var/opt/oracle/oratab** file that have the third field specified as **Y** (yes). In addition, this script will startup the orasrv process and the msql daemon.

Create a file named **dbora** in the **/etc/init.d** directory.

Confirm that there are entries like the following at the end of the file (be sure to give the full path of the **dbstart** utility):

ORACLE_HOME=/opt/oracle
ORA_HOME=/opt/oracle
ORA_OWNER=oracle
su - $ORA_OWNER -c $ORA_HOME/bin/dbstart &
su - $ORA_OWNER -c $ORA_HOME/bin/orasrv
/usr/local/Minerva/bin/msqld &

If these entries do not exist, you need to add them to the file.

Link this file to files in **/etc/rc2.d** by entering the following:

 # ln -s /etc/init.d/dbora /etc/rc2.d/K98dbora

 # ln -s /etc/init.d/dbora /etc/rc2.d/S99dbora

2.4.1.4 ORACLE Database Shutdown for SunOS 4.1.3. Before SunOS can be shutdown, all ORACLE databases should be properly closed. The **dbshut** command can be run interactively to shutdown all the ORACLE databases listed in the **/etc/oratab** file that have **Y** (yes) in the third field.

The entries in the **/etc/oratab** file have the syntax:

$ORACLE_SID:$ORACLE_HOME:<N|Y>

To interactively shutdown the databases, the superuser can enter the command:

su - oracle -c /usr/oracle/bin/dbshut

The ORACLE databases can also be shutdown within the SQL*DBA tool. The SQL*DBA is a tool for helping database administrators manage and monitor a database. SQL*DBA may be used interactively or an SQLDBA command line may be executed at the operating system level. Refer to the *ORACLE RDBMS Database Administrator's Guide* for detailed information on the SQL*DBA tool.

2.4.1.5 ORACLE Database Shutdown for Solaris 2.3. Before Solaris can be shutdown, all ORACLE databases should be properly closed. The **dbshut** command can be run interactively to shutdown all the ORACLE databases listed in the **/var/opt/oracle/oratab** file that have **Y** (yes) in the third field.

To interactively shutdown the databases, the superuser can enter the command:

su - oracle -c $ORACLE_HOME/bin/dbshut

2.4.1.6 ORACLE Process Architecture. An ORACLE database is comprised of four background processes and the System Global Area (SGA). The database is known as an instance, and only one instance may point to one database at any given time. The four background processes are described below:

a. **LGWR - Redo Log Writer**.

(1) Writes SGA redo log buffer blocks to the redo log when:

(a) A COMMIT occurs,
(b) The redo log buffer pool fills, or
(c) The DBWR requests it.

(2) Piggybacks COMMITs to achieve an average rate of less than one I/O per commit.

(3) Allows archiving of logs for roll-forward recovery.

b. **DBWR - Database Writer.**

(1) Writes "used blocks" from the SGA database buffer as free ones are needed.

(2) Keeps in memory the data most needed.

(3) Piggybacks transactions to achieve fewer I/Os per data block.

(4) Writes to database files those buffers for which changes have been written in the redo log.

c. **PMON - Process Monitor.**

(1) Processes recovery (i.e., cleans up aborted processes).

(2) Rolls back uncommitted transactions by releasing locks and freeing up SGA resources.

d. **SMON - System Monitor.**

(1) Instance recovery that cleans up the database.

(2) Temporary segment reclamation of unused temporary segments.

2.4.1.7 <u>ORACLE Storage Requirements</u>. Tables 2-II and 2-III list the storage requirements for the ORACLE tools used by the DSRS.

Table 2-II. ORACLE Tool Storage Requirements for SunOS 4.1.3

Oracle Tools	Disk Space Mb	User #1 Memory Mb	Additional User Memory Mb
RDBMS/TPO	45.14	6.840	.752
SQL*Plus	4.97	1.917	.605
SQL*Loader	X[1]	1.760	.560
Pro*C	3.57	3.359	.959
Database Logfile	12.5	X[1]	X[1]
Sub Total	66.18	13.876	.876

X[1] Not Available

Table 2-III. ORACLE Tool Storage Requirements for Solaris 2.3

Oracle Tools	Disk Space Mb	User #1 Memory Mb	Additional User Memory Mb
RDBMS/TPO	59.85	6.848	.036
SQL*Plus	7.30	2.081	.061
SQL*Loader	X[1]	1.922	.024
Pro*C	5.28	3.135	.054
Database Logfile	12.5	X[1]	X[1]
Sub Total	84.93	13.986	.175

X[1] Not Available

2.4.1.8 **Database File Maintenance**. Physical database files are mapped to logical divisions of a database called tablespaces. A tablespace must be made up of at least one file; however, it can contain multiple files. Because of this close relationship between database files and tablespaces, most file maintenance is done by invoking SQL statements referencing tablespaces. For instance, the SQL statements, **CREATE TABLESPACE** or **ALTER TABLESPACE**, can be used to:

a. Specify the database file(s) to be mapped to a tablespace,

b. Add a file to a tablespace, and

c. Rename a file in a tablespace (using either ALTER TABLESPACE or ALTER DATABASE).

Database files cannot be dropped or enlarged; however, another file can be added to a tablespace to enlarge the size of the existing tablespace.

2.4.1.9 **Tablespace Maintenance.** The tablespace is the primary logical division of a database. It is the unit used for space allocation, on-line availability, and recovery. The following subsections describe the tablespace maintenance functions: monitoring space usage within a tablespace, increasing the size of a tablespace (by adding a file), setting or changing default storage parameters associated with the tablespace, renaming files in a tablespace, taking tablespaces off-line and on-line, and dropping tablespaces.

NOTE: The SQL statements executed to maintain tablespaces (CREATE TABLESPACE, ALTER TABLESPACE, and DROP TABLESPACE) require Database Administrator (DBA) privileges. Also, some of the options of these statements may only be executed while a tablespace is off-line.

2.4.1.9.1 Monitoring Space Usage in a Tablespace. The following data dictionary views allow users to monitor how much space is allocated, used, or free in a tablespace:

a. DBA_SEGMENTS

b. DBA_EXTENTS

c. DBA_FREE_SPACE

2.4.1.9.2 Enlarging a Tablespace. Adding a new database file to a tablespace will increase the size of that tablespace. This new database file will always be a subsequent file for the tablespace, as the first file is specified in the CREATE TABLESPACE command. To add a new database file:

a. The database should be open; the tablespace can be on-line or off-line.

b. Enter SQL*DBA and connect to the database as a DBA.

c. Use the following syntax to add the file:

ALTER TABLESPACE *tablespace*
ADD DATAFILE *filespec* **SIZE 10M**

where *tablespace* is the name of the tablespace where the file will be added and *filespec* is the database file being added. The syntax for *filespec* is operating system-dependent. Be sure to use a new filename; if you use the name of an existing database file, results are not predictable.

2.4.1.9.3 Setting Default Storage Parameters for a Tablespace. One characteristic of a tablespace is that it sets the default storage parameters and limits for tables or indexes created in the tablespace. When creating the tablespace, storage parameters can be set using the storage options for the **CREATE TABLESPACE** statement.

2.4.1.9.4\ Changing Default Storage Parameters. To alter the default storage parameters for a tablespace, the following SQL statement (note the keywords DEFAULT STORAGE) can be executed:

ALTER TABLESPACE *tablespace*
DEFAULT STORAGE (
[INITIAL *bytes*] [**NEXT** *bytes*]
[MINEXTENTS *int*] [**MAXEXTENTS** *int*]
[PCTINCREASE *pct*] **)**

2.4.1.9.5 Setting User Quotas for Tablespace Usage. When granting a user RESOURCE privilege to a tablespace, a limit can be specified for the number of bytes that the user may use in that tablespace. For example, the following SQL statement:

GRANT RESOURCE (1300) ON TSPACE_TWO TO NANCY

allows the user NANCY to use 1300 bytes in the tablespace TSPACE_TWO. The same statement can be used to impose or alter a quota. If the user has already used more space than the new quota allows, then the user cannot consume more space until his/her storage drops below the new limit.

2.4.1.9.6 Renaming Files in a Tablespace. Users may want to rename files in a tablespace in order to relocate files to different devices or to use files of different sizes. To rename database files use the following SQL statement:

ALTER TABLESPACE *tablespace*
RENAME DATAFILE *file1* **TO** *file2*

where *tablespace* is the name of the tablespace where the file exists, *file1* is the file to be renamed, and *file2* is the new name of the file.

2.4.1.9.7 Taking a Tablespace Off-line. A tablespace may be taken off-line for any of the following reasons: to make the data in the tablespace (temporarily) unavailable, to free the disk space used by the tablespace for another purpose, or to perform tablespace maintenance (such as adding a file for performing backups). A tablespace does not need to be off-line when adding a file for a backup. To take a tablespace off-line follow these steps:

a. Check that the database is open.

b. Connect to the database using the keyword INTERNAL.

c. Verify that there are no active rollback segments in the tablespace by querying the data dictionary view DBA_ROLLBACK_SEGS. If there are active segments, the tablespace cannot be taken off-line until the rollback segments have been dropped.

d. Use the following syntax:

ALTER TABLESPACE *tablespace*
OFFLINE [NORMAL | IMMEDIATE]

Use NORMAL to indicate that the tablespace should only go off-line when all users are finished, or use IMMEDIATE to indicate that it should go off-line in any case.

A tablespace will remain off-line (through instance startup and shutdown) until it is brought back on-line.

2.4.1.9.8 Bringing a Tablespace On-line. In order to access data within a tablespace, it must be on-line. Follow these steps to bring an off-line tablespace on-line:

a. Check that the database is mounted but shut.

b. Connect using the keyword INTERNAL.

c. Enter the SQL statement:

ALTER TABLESPACE *tablespace* **ONLINE**

2.4.1.9.9 Dropping Tablespaces. Tablespaces may be dropped, even if they have table data in them. To drop an on-line tablespace, use the syntax:

DROP TABLESPACE *tablespace* [**INCLUDING CONTENTS**]

If a tablespace containing data is to be dropped, the option **INCLUDING CONTENTS** must be used for it to be dropped successfully.

2.4.1.10 Rollback Segment Maintenance. Rollback segments contain information required for read consistency and to undo changes made by transactions that rollback. Most databases have multiple rollback segments.

The original rollback segment, called SYSTEM, is automatically created as a part of the **CREATE DATABASE** statement. It is created in the SYSTEM tablespace and its initial size is defined by the default storage parameters used by the SYSTEM tablespace.

If there are multiple tablespaces, there must be at least one rollback segment in addition to the SYSTEM rollback segment.

2.4.1.10.1 Monitoring Rollback Segments. To monitor how much space rollback segments are using, query the data dictionary views DBA_ROLLBACK_SEGS and DBA_SEGMENTS. The SQL*DBA command **MONITOR ROLLBACK** can also be used to see information about rollback segments.

For example, to view the current rollback segments of the database, query the DBA_ROLLBACK_SEGS view of the data dictionary with the following statement:

SELECT SEGMENT_NAME, TABLESPACE_NAME, STATUS
FROM SYS.DBA_ROLLBACK_SEGS

2.4.1.10.2 Creating Rollback Segments. To create additional rollback segments for a database, use the **CREATE ROLLBACK SEGMENT** command. The **CREATE ROLLBACK SEGMENT** command has the syntax:

CREATE [**PUBLIC**] **ROLLBACK SEGMENT** *rs_name*
[**TABLESPACE** *ts_name*]
[**STORAGE** (storage)]

where *rs_name* is the name of the new rollback segment and *ts_name* is the name of the tablespace to which it will be assigned. If the keyword PUBLIC is used, a public rollback segment is created.

A rollback segment is not in use unless it is acquired at instance startup; thus, a rollback segment that has just been created is not used until an instance shuts down and restarts, acquiring the segment either as a public segment or through naming it in the INIT.ORA file.

New rollback segments can be created in any tablespace, but it is best to create rollback segments in tablespaces that will remain on-line; a tablespace cannot be taken off-line if it contains an active rollback segment. (The data dictionary view DBA_ROLLBACK_SEGS lists all active rollback segments with the status INUSE.)

2.4.1.10.3 Specifying Storage Parameters for Rollback Segments. When a rollback segment is created, storage parameters can be specified for the segments's extent. The syntax of the storage parameters is:

STORAGE (
[**INITIAL** *bytes*] [**NEXT** *bytes*]
[**MINEXTENTS** *int*] [**MAXEXTENTS** *int*]
[**PCTINCREASE** *pct*] **)**

In general, all extent for rollback segments should be the same size by specifying identical values for INITIAL and NEXT and setting the PCTINCREASE to 0.

2.4.1.10.4 Altering Rollback Segment Storage Parameters. The rollback segment storage parameters can be changed for a specific rollback segment with the **ALTER ROLLBACK SEGMENT** command:

ALTER [**PUBLIC**] **ROLLBACK SEGMENT** *rs_name*
STORAGE (
[**NEXT** *integer*]
[**MAXEXTENTS** { *integer* | NULL}]
[**PCTINCREASE** *integer*])

However, the **ALTER ROLLBACK SEGMENT** command only affects future extent used by the rollback segment.

2.4.1.10.5 Activating Rollback Segments. When an instance starts up, it attempts to acquire a number of rollback segments in the ratio of the values of INIT.ORA parameters TRANSACTIONS and TRANSACTIONS_PER_ROLLBACK_SEGMENT, rounded up to the next integer value. The instance first acquires all private rollback segments named in the ROLLBACK_SEGMENTS parameter. If ROLLBACK_SEGMENTS names N private rollback segments and the instance requires more than N rollback segments, the instance tries to acquire the following number of public rollback segments:

$$\text{Number of public rollback segments the instance requires} = \frac{\text{TRANSACTIONS}}{\text{TRANSACTIONS\_PER\_ROLLBACK\_SEGMENT}} - N$$

Depending on the type of rollback segment that was created, public or private, there are two ways of making it available for use.

Once a **public** rollback segment is created, it cannot be used until the instance shuts down and the instance restarts. (NOTE: Restarting an instance that uses public rollback segments does not ensure that the instance uses any particular public rollback segment.)

Once a **private** rollback segment is created, it cannot be used until the instance shuts down, the name of the new rollback segment is added to the ROLLBACK_SEGMENTS parameter in the INIT.ORA file of the instance, and the instance restarts. (NOTE: The instance acquires all of the named rollback segments even if ROLLBACK_SEGMENTS lists more rollback segments than the ratio, TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT.)

2.4.1.10.6 Dropping Rollback Segments. A rollback segment may be dropped because: it becomes too fragmented; has many extents; the instance using it no longer accesses the database; it needs to be recreated as public or private; or it needs to be relocated in a different tablespace.

A rollback segment cannot be dropped if an instance has acquired it and is using it. To drop a rollback segment follow these steps:

a. Check that the database is open.

b. Connect using a DBA username.

c. Make sure the rollback segment to be dropped is not in use by any instance by querying the data dictionary view DBA_ROLLBACK_SEGS.

d. Enter the SQL statement:

DROP [**PUBLIC**] **ROLLBACK SEGMENT** *name*

2.4.1.10.6.1 Dropping Private Rollback Segments. The INIT.ORA parameter ROLLBACK_SEGMENTS determines all private rollback segments that the instance attempts to acquire when it starts. To drop a private rollback segment that is currently in use, first shut down the instance, remove the rollback segment name from the ROLLBACK_SEGMENTS parameter, and start the instance again. Then, the private rollback segment(s) can be dropped.

2.4.1.10.6.2 Dropping Public Rollback Segments. To drop a public rollback segment that is acquired at startup time, the instance must be prevented from using it. A public rollback segment that you want to drop can be deactivated by creating and substituting it with private rollback segments. The instance must be shut down and restarted with at least as many private rollback segments as the ratio, TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT; then the public rollback segment(s) can be dropped.

2.4.1.11 Compressing Extents. Extents can be compressed with the ORACLE Export Utility. To compress extents for all data in a database, a full database export must be performed with the COMPRESS flag set to Y. After a full database export has been performed, the database must be recreated and a full database import must be performed.

The ORACLE Import utility imports a table into its original tablespace, using the original storage parameters. If you select the "compress extent" option at export time, then the storage parameters for large tables are adjusted so that all data imported for a table is placed into its initial extent.

2.4.2 Report Inventory.

2.4.2.1 DSRS Reports. The Supervisor/Librarian can generate reports on the data in the system. These reports are predefined to reduce the complexity of generating reports. The reports furnish historical information of the user's audit information and the database contents. The Supervisor/Librarian invokes the DSRS Librarian Report Tool to specify which reports to generate. The reports are generated using Borland ReportSmith for Windows Runtime Viewer. The predefined reports may be customized with the Borland ReportSmith for Windows.

Borland ReportSmith for Windows Runtime Viewer is delivered with the DSRS. Borland ReportSmith for Windows may be purchased by sites to customize their reports.

2.4.2.2 ORACLE Reports. ORACLE errors, due to the failure of one of the background processes, are recorded in trace (dump) files generated by the process. These files are found in the location specified by the INIT.ORA parameter, BACKGROUND_DUMP_DEST, and give some indication of why the process may be failing.

2.4.2.3 SunOS and Solaris Reports. The SunOS and Solaris environments have an error logging daemon, **syslogd**, that is used by many system facilities to record error messages whenever an unusual event occurs. These messages are written to the file **/var/adm/messages**. This file is available for viewing errors that occur on the system.

2.4.2.4 DSRS Logs. DSRS writes notification messages (extraction requests) and error messages to the **pc_server.log**, **lib_server.log**, and **dsrs_server.log** files. These files are stored in the **/var/adm/dsrs** directory.

The DSRS logs will become very large if they are not deleted on a regular basis. It is recommended that after each Level 0 dump is performed, the **pc_server.log**, **lib_server.log**, and **dsrs_server.log** files be deleted from the **/var/adm/dsrs** directory. However, these files should not be deleted without being backed up to tape first. These files will be automatically created, if they do not already exist, when the servers are executed.

A script file **/dsrscm/bin/cleanup_logs** has been provided to automate the deletion of the DSRS log files. The script file **/dsrscm/bin/cleanup_logs** should be inserted at the bottom of the monthly level 0 dump script file, or executed by root after level 0 dumps are performed.

DSRS also creates a file each time a tape copy or hard copy extraction is requested. The file created has the filename ***[USERID] [Timestamp].EXT*** where ***[USERID]*** is replaced with the userid for the user who requested the extraction and ***[Timestamp]*** is replaced with the date and time of extraction. These files are stored in the **/var/adm/dsrs** directory and contain information about the extraction including the filenames for the files associated with the RA for which the user is requesting an extraction. If the extraction request is from a remote DSRS site, then the user's site ID who is requesting the extraction is preappended to the filename.

2.4.3 DSRS Executable Storage Requirements. Table 2-IV lists the storage requirements for the DSRS executables.

Table 2-IV. DSRS Executable Storage Requirements

EXECUTABLE	UNIX (KB)
DSRS	6752
DSRS	480
DSRSLIB	776
pc_server	1744
pc_server.sol23	2851
lib_server	1736
lib_server.sol23	2852
dsrs_server	1632
dsrs_server.sol23	2712

2.5 COMMUNICATIONS REQUIREMENTS. The communications path between remote sites will take place over Defense Data Network (DDN)/Internet for the UNIX system. Communications over DDN will be supported via Transaction Control Protocol/Internet Protocol (TCP/IP) Sockets. Dial-in users will require terminal emulation communications software that supports Serial Line Internet Protocol (SLIP) or Point-to-Point Protocol (PPP). The medium for dial-up terminals is a modem with voice-grade telephone lines.

2.5.1 Graphic Overview. The hardware and software environment will conform to the above listed minimum requirements. A graphical overview of the DSRS communications paths is displayed in Figure 2-5.

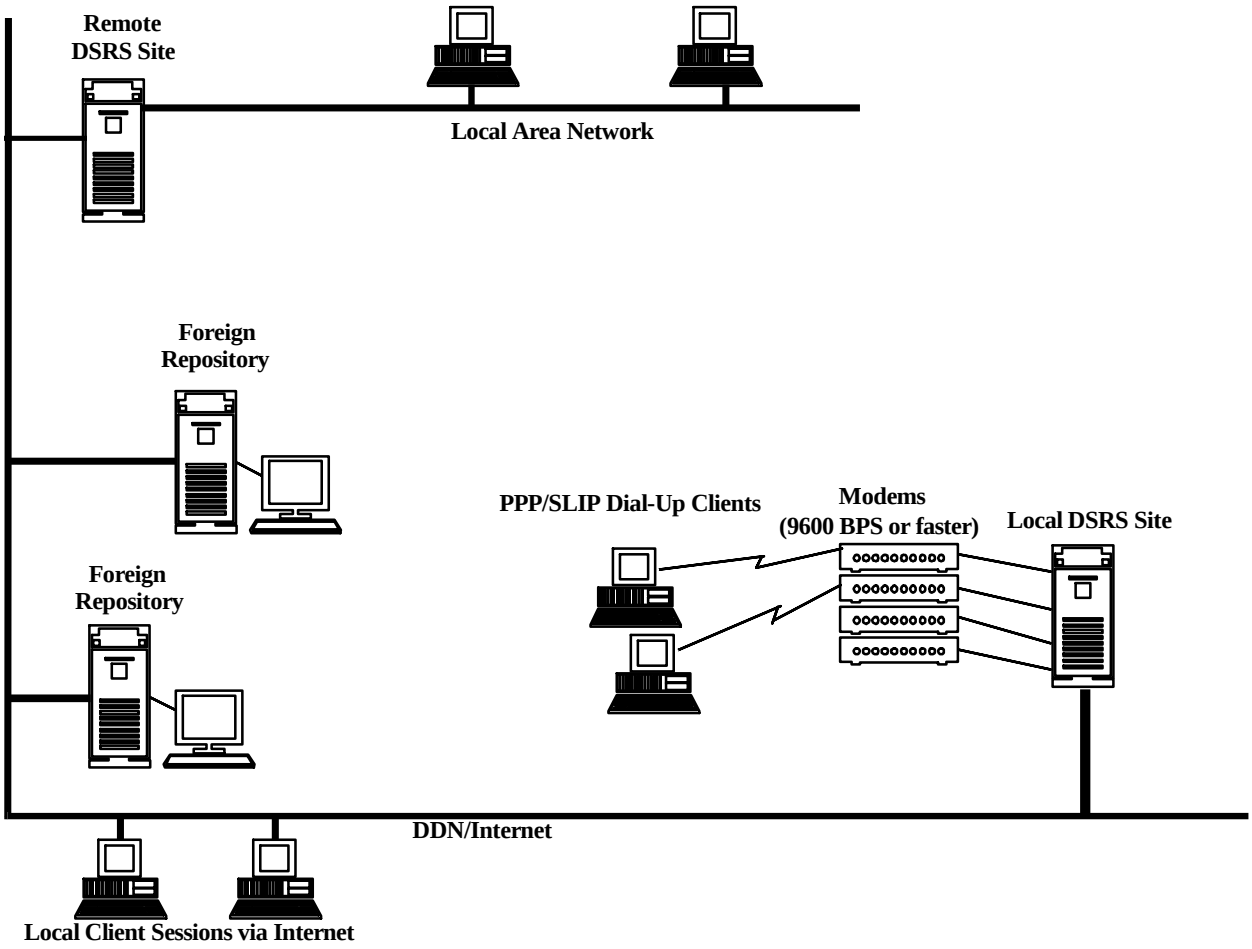


Figure 2-5. DSRS Communications Overview

2.5.2 Hardware. DSRS site computers require appropriate Internet connections to be able to perform remote extractions. This may include Ethernet hardware, routers, and appropriate assignment of Internet addresses.

2.5.3 Software. Software operating on hardware systems described in Section 5.1 of the *Functional Description of the DSRS* will conform to the requirements identified in the previous sections. Users may use various TCP/IP software packages to access the DSRS. Network connections to the Internet will require appropriate TCP/IP networking software.

2.5.4 Modems. A modem may be used from a PC to connect to the MS-Windows User Tool. Using a modem to connect to the DSRS requires hardware and software to be installed on the calling end and on the server which is the receiving end. The user needs a recommended minimum 9600 bps modem installed on the calling PC. The TCP/IP software on the user's PC needs a PPP/SLIP option. The server needs a recommended minimum 9600 BPS modem for users to establish a PPP/SLIP connection with a modem. The TCP/IP software on the server needs a PPP/SLIP option. The Morning Star PPP V1.4.1 software has been installed and tested on a SunOS 4.1.3 environment. This

software will provide the necessary PPP protocol to allow dial-in users using PPP-supported software to use both the Windows-based DSRS User Tool and the DSRS Librarian Tool.

Using the modem to connect to the server allows the PC access to the DSRS and the Internet. Access to the Internet is required for proper operation of the DSRS. The PPP/SLIP connection to the server creates the TCP/IP stack necessary for Internet communication.

2.5.5 Interfaces. The communication architecture is illustrated in Figure 2-5.

2.5.5.1 TCP/IP Asset Transfer Interface. The DSRS component transfer is an interaction between two sites (DSRS or foreign) via an application layer protocol transported via TCP/IP. The protocol implements a client/server architecture in which the local DSRS User Tool becomes a client to the Interoperability Server of an owning site. The Interoperability component transfer protocol will provide the basic set of operations for gathering data, transmitting it, and using it properly on the receiving site. This protocol will operate over the Internet and/or local TCP/IP networks. Remote extraction is automated by the DSRS User Tool through the use of the extract command. A user's extract request for remote RAs generates a client transaction to the cooperating site's server; the local DSRS User Tool will communicate with the cooperating site's server. The Interoperability Server located at the remote site will process extract requests and transmit components and extract status to the client.

2.5.5.2 DSRS-to-ORACLE Interface. Since DSRS uses ORACLE as its underlying database, a communications interface exists between them. The DSRS communicates with ORACLE using SQL and the ORACLE database interface. This communication is local to one computer system; i.e., the DSRS and ORACLE will execute on the same platform.

2.5.5.3 User Terminal Interface. Users will connect with the DSRS using PPP/SLIP dial-up MS Windows connections, sessions over the Internet, or X sessions over the Internet. The interface between the user's terminal and the DSRS is provided in large part by the underlying operating system and networking software. X sessions may be executed over a TCP/IP network such as the Internet.

2.6 SECURITY. Based upon the perceived threats, risks, and goals of the DISA Software Reuse Program, the security policy for the DSRS is to provide controlled access protection (C2 functionality). Detailed information concerning security requirements for controlled access protection are included in Section 6 of the *Functional Description for the DSRS* and are derived from the following three basic objectives:

a. Protection and control over who can login to the system.

b. Mechanisms that will enable the DSRS to make decisions regarding access to resources based upon the expressed wishes of its users (with no assurance that concerted, malicious actions cannot circumvent these mechanisms).

c. The capability of generating a reliable log of user actions and to guarantee its correctness.

2.6.1 Logon Protection and Control. Protection and control over who can log into the DSRS is accomplished through a combination of procedural and automated security means. The first step a potential user of the DSRS must accomplish is the completion of an Account Request Form (ARF) that contains a User Non-Disclosure Agreement and a section to be completed by a Government Program Manager/Representative. Both the user and the manager/representative are required to sign the form before an account on the DSRS will be assigned to the user.

Identification and authentication is assured in the DSRS by prompting the user to supply a login name and password. Identification is implemented by asking for a login name which is associated with the user's identity. The DSRS checks this name against its list of authorized users. The system then asks the user to provide a password to authenticate that the user is who he or she claims to be.

2.6.2 Discretionary Access Control. Discretionary Access Control (DAC) is a means of restricting access to named objects based upon the identity of subjects and/or groups to which they belong. DAC is accomplished in the DSRS by assignment of group identifiers to site IDs, user IDs and RA IDs. A group ID is a tag identifying an abstract collection of sites, users, and assets and conveying certain rights and privileges to the members of that collection. Another mechanism in place to control user access in the DSRS is the assignment of specific user types, according to the functions a user will need to perform in the system. The following DSRS user types will be assigned to determine the level of access granted to the user:

a. **Non-User** A person who is identified in the system as being a point-of-contact for an RA. A non-user will not be able to execute the system.

b. **Programmer** A person who will be able to execute the User Tool to identify RAs.

c. **Librarian** A person who maintains the RA Catalog. This user will be able to execute all of the tools associated with the system.

d. **Supervisor** A person who maintains the system catalog. This user will be able to execute all of the tools associated with the system. In addition to basic Librarian privileges, this user will be able to: delete RAs, add and delete domains, modify local site information, and maintain user accounts for system supervisors.

2.6.3 Audit. The audit requirements are that the system have the capability to create, maintain, and protect, from modification or unauthorized access or destruction, an audit trail of accesses to the objects it protects. The DSRS was designed to allow authorized operations and security personnel to generate a number of reports concerning the interactions between subjects (users) and objects (data) in the system. Below is the list of reports and log files supported by the DSRS:

a. pc_server.log

b. lib_server.log

c. dsrs_server.log

d. Usage Log

Refer to Section 2.4.2.4 of this document for details on the files (items a, b, and c) listed above.

While not all of the above listed reports were specifically designed with security in mind, the DSRS Security Administrator is able to obtain a complete view of activities on the DSRS system. These logs permit DSRS system audits. Examination and maintenance of audit reports will be provided in the *DSRS Security Plan* and the *DSRS Trusted Facility Manual*.

ORACLE audit capabilities are implemented for the following commands: delete, insert, update, grant, and all sessions. Information about auditing database usage can be found in the *ORACLE7 Server Administrator's Guide*. In addition, Appendix D lists several auditing tables and recommended select statements for viewing the audit tables.

2.7 DSRS ENVIRONMENT VARIABLES AND UNIX SHELLS. When a user logs into the SunOS or Solaris environment, he/she is put into one of three shells: the captive shell, the C shell or the Bourne shell. The UNIX shell is simply a program that parses commands and passes control to other programs that make up the operating system. The shells differ in the functions they perform and the command syntax they accept, but all pass control to the UNIX programs for processing except for commands processed by the shell itself. The captive shell does not allow the user direct access to the SunOS or Solaris environment, therefore, this shell should be used by all DSRS X/Motif users other than Librarians and Supervisors.

The default shell, defined during account creation will be made available to the user. Unless specified by the user, it is recommended that the captive shell be assigned to each user. Each shell presents a prompt: **Enter Selection:** for the captive shell; a percent sign (**%**) for the C shell. Each shell is briefly described in the following subsections. The DSRS will not run in the Bourne shell.

2.7.1 Captive Shell.

a. The captive shell prompt is the text: **Enter Selection:**.

b. Captive shell users have all their environment variables defined in the **/bin/captive** script and the **/dsrscm/bin/setup.csh** script.

c. Captive shell users are restricted to accessing only the selections available in the menus and they have no access to the SunOS or Solaris.

2.7.2 C Shell.

a. The default C shell prompt is a percent sign (**%**).

b. C shell users have two methods for defining the user environment. The first method uses the hidden file **.login**, located in the parent directory. During login, users can change the default values of these variables, or add to them, by including the appropriate definitions in the **.login** file. The second method for defining the user

environment applies only to those commands interpreted by the C shell. The hidden file **.cshrc**, located in the parent directory, governs the C shell environment. Entries in the **.cshrc** file define shell variables. By default, the new process receives the characteristics found in both **.login** and **.cshrc**.

c. To define the DSRS environment variables, described in Section 2.7.4, the following lines must be placed in the user's **.cshrc** file.

```
setenv ORACLE_HOME [ORACLE_HOME]
set path = ($PATH $ORACLE_HOME/bin)
setenv ORACLE_SID [ORACLE_SID]
source /dsrscm/bin/setup.csh
```

The previous commands require [ORACLE_HOME] and [ORACLE_SID] to be replaced with the value to which each variable is set.

d. To define the ORACLE environment variables, the following lines must be placed in the user's **.login** file.

```
setenv ORAENV_ASK=NO
source $ORACLE_HOME/bin/coraenv
unset ORAENV_ASK
```

e. To invoke the C shell environment the user can enter the command:

```
/bin/csh
```

f. The C shell environment definition file, **.cshrc**, usually consists of two types of commands, **setenv** and **alias**. The **setenv** command defines C shell *variables*. A *variable* is a named location in which to store text that the C shell remembers. The command **alias** redefines command names.

g. A user may execute the **.cshrc** file the C shell, without having to log off and log in by entering the command:

```
source $HOME/.cshrc
```

2.7.3 DSRS Environment Variables. The following are a list of some of the DSRS variables which are set when the **/dsrscm/bin/setup.csh** script is executed:

a. DSRS_BASE

b. DSRS_LIBRARIAN

c. DSRS_TESTDB

d. DSRS_USE

e. DSRS_UTIL

f. UIDPATH

g. USR_MOTIF_DIR

2.8 GENERAL SYSTEM ADMINISTRATION - SunOS 4.1.3. Table 2-V lists general system administration activities, a description of each, frequency of activity, and a paragraph reference where further information on the activity can be found.

Table 2-V. General System Administration Activities for SunOS 4.1.3

ACTIVITY	DESCRIPTION	FREQUENCY	PARAGRAPH REFERENCE
Cleaning File Systems	Steps to clean the critical file systems.	as required	2.8.3
Create SunOS Accounts	Add user accounts to the SunOS.	as required	2.8.4
Create SunOS Groups	Add user accounts to SunOS groups.	as required	2.8.5
Configuring NIS	Steps to configure NIS	as required	2.8.21
Configuring WWW Server	Steps to configure WWW Server	as required	2.8.22
Debug DSRS Problems	Steps to debug the DSRS when executables are not working.	as required	2.8.6
Debug ORACLE Problems	Steps to debug Oracle when there are database problems.	as required	2.8.7
Debug SunOS Problems	Steps to debug the SunOS when there are operating system problems.	as required	2.8.8
Formatting a Disk	Steps to format a disk.	as required	2.8.16
Making a New File System	Steps to create a new file system.	as required	2.8.18
Mounting the File Systems	Steps to automatically mount file systems.	as required	2.8.19
ORACLE Backups	Create database export files.	daily Monday-Friday	2.8.1
Partitioning a Disk	Steps to partition a disk.	as required	2.8.17
Reconfiguring the Kernel	Steps to reconfigure and build a kernel.	as required	2.8.20
Restore Backups for ORACLE	Recover database information from export file.	as required	2.8.9
Restore Backup for SunOS	Restore SunOS files from backup tapes.	as required	2.8.10
Saving ORACLE Audit Data	Save ORACLE Audit Data	monthly	2.8.23
Shutdown ORACLE	Shutdown the Oracle database.	as required	2.4.1.4
Shutdown SunOS	Shutdown the SunOS.	as required	2.8.13
Startup ORACLE Database	Startup the Oracle database.	as required	2.4.1.2
Startup SunOS	Startup the SunOS.	as required	2.8.15
SunOS Backups	Perform backups on the SunOS environment.	daily Monday-Friday	2.8.2

2.8.1 ORACLE Backups. Backups may be performed for the ORACLE database information with the ORACLE Export utility. The Export utility writes data from the ORACLE database to operating system files in an ORACLE binary format. These files may be read into an ORACLE database to

restore information, using the ORACLE Import utility. Refer to Section 2.8.9 for information on the ORACLE Import utility.

The Export utility can be executed by either command-line or interactive methods. The command-line method is recommended for backing up the ORACLE database because it can be placed in a command file and be run in a background process. Three different modes can be used during the Export utility: Table, User, and Full Database. Table 2-VI describes each mode and Table 2-VII lists the objects exported for each of the three modes.

Table 2-VI. Export Modes

MODE	DESCRIPTION
Table	This mode allows the user to specify which tables to export that they own, rather than all of the tables. A DBA can qualify the tables by specifying who owns them. The default is to export all tables belonging to the user doing the export.
User	This mode allows the user to export all objects (such as, tables, data, grants, and indexes) belonging to one user. A DBA exporting in user mode can export all objects belonging to a specified list of users.
Full Database	Only DBAs can export in this mode, which exports all database objects except those owned by SYS.

Table 2-VII. Objects Exported for Each Mode

Table Mode	User Mode	Full Database Mode
table definitions	table definitions	table definitions
table data	table data	table data
owner's grants	owner's grants	grants
owner's indexes	owner's indexes	indexes
table constraints	table constraints	table constraints
table triggers	table triggers	all triggers
	clusters	clusters
	database links	database links
	views	views
	private synonyms	all synonyms
	sequences	sequences
	snapshots	system privileges
	snapshot logs	tablespace definitions

Table Mode	User Mode	Full Database Mode
	stored procedures	tablespace quotas
		rollback segment definitions
		system audit options
		snapshots
		snapshot logs
		stored procedures
		profiles
		roles
		user definitions

ORACLE backups are performed in **Full Database** mode with all table GRANTS and data. An example of the command-line for a Full Database export follows:

exp userid=system/manager full=Y file=dba.dmp grants=Y indexes=Y

The command-line method allows parameters to control an export. The parameters are detailed in Table 2-VIII.

Table 2-VIII. Control Parameters

Parameter	Default Value	Description
userid	undefined	The username and password (separated by a slash) of the user performing the export.
full	N	A flag to indicate whether to export the entire database. Specify full=Y to export in full database mode.
file	expdat.dmp	The name of the output file created by Export.
grants	Y	A flag to indicate whether to export grants.
indexes	Y	A flag to indicate whether to export indexes.

Refer to the *ORACLE7 Server Utility User's Guide* for a complete description and use of all Export parameters.

2.8.2 SunOS Backups. Backups are one of the most crucial system administration functions. A procedure for regular scheduled backups of all file systems is needed for two major reasons: to ensure file system integrity against possible system crash, and to ensure user files against accidental deletion.

If file systems are backed up as scheduled, then the corrupted files can be restored to a reasonably recent state. In SunOS terminology, the word *dump* is often used to mean "back up".

The dump command backs up files on a per-file-system basis. The dump command allows two kinds of dumps: full dumps and incremental dumps. With levels, ranging from 0 to 9, the user may specify whether all files shall be dumped or only those that were altered since the last lower level dump.

A Level 0 dump is a full dump (a backup of the contents of an entire file system). Levels 1 to 9 dumps, perform an incremental dump (a backup of only files that have changed since the last dump of a lower level).

2.8.2.1 Backup Strategies. Table 2-IX illustrates an **EXAMPLE** of a backup plan where users are doing file-intensive work. It assumes a theoretical month that begins on Friday and consists of four, five-day work weeks. A Level 0 dump writes every file in the specified file system to the dump tape. A Level 9 dump writes every file in the specified file system to the dump tape that has changed since the last Level 8 or lower dump. In practice, at least one more Level 9 dump will be necessary, depending upon the number of days in the month. Also, an end-of-month backup may be scheduled on the last Friday or, if applicable, the last weekend of the month.

Table 2-IX. Schedule of Backups

Directory	Date	Level	Tape Name
/	end-of-month	Level 0	*n* tapes
/usr	end-of-month	Level 0	*n* tapes
/export	end-of-month	Level 0	*n* tapes
/home	end-of-month	Level 0	*n* tapes
/ /usr /export /home	1st Friday	Level 5	Tape A
/ /usr /export /home	1st Monday	Level 9	Tape B
/ /usr /export /home	1st Tuesday	Level 9	Tape C

Directory	Date	Level	Tape Name
/ /usr /export /home	1st Wednesday	Level 9	Tape D
/ /usr /export /home	1st Thursday	Level 9	Tape E
/ /usr /export /home	2nd Friday	Level 5	Tape F
/ /usr /export /home	2nd Monday	Level 9	Tape B
/ /usr /export /home	2nd Tuesday	Level 9	Tape C
/ /usr /export /home	2nd Wednesday	Level 9	Tape D
/ /usr /export /home	2nd Thursday	Level 9	Tape E
/ /usr /export /home	3rd Friday	Level 5	Tape G
/ /usr /export /home	3rd Monday	Level 9	Tape B
/ /usr /export /home	3rd Tuesday	Level 9	Tape C

Directory	Date	Level	Tape Name
/ /usr /export /home	3rd Wednesday	Level 9	Tape D
/ /usr /export /home	3rd Thursday	Level 9	Tape E
/ /usr /export /home	4th Friday	Level 5	Tape H
/ /usr /export /home	4th Monday	Level 9	Tape B
/ /usr /export /home	4th Tuesday	Level 9	Tape C
/ /usr /export /home	4th Wednesday	Level 9	Tape D
/ /usr /export /home	4th Thursday	Level 9	Tape E

With this plan, *n* tapes (the number of tapes needed for a full backup of /, /usr, /export, /home) will be used plus eight additional tapes for the incremental dumps. This plan assumes that each incremental dump uses one tape. If large file systems exist, or if there are more than those file systems listed, then more tapes may be needed.

This is how the plan will work:

a. At the end of the month, a full backup (Level 0) of all file systems is created. These tapes need to be saved for a year.

b. On the first Friday of the month, a Level 5 dump of all file systems is created which copies all files changed since the previous lower-level dump; in this case, the Level 0

dump that was performed at the end of the month. Tape A is used for this dump, saved for the month, then used for the first Friday of the next month.

c. On the first Monday of the month, Tape B is used to perform a Level 9 dump of all file systems. The dump copies all files changed since the previous lower level dump; in this case, the Level 5 dump that was performed on Friday. Then, Tape B must be stored until the following Monday, when it will be used again.

d. On the first Tuesday of the month, Tape C is used to perform a Level 9 dump of all file systems. Again, the dump copies all files changed since the last lower level dump (Friday's Level 5 dump).

e. Wednesday and Thursday Level 9 dumps must be created on Tapes D and E, respectively.

f. At the end of the week, Tape F is used for a Level 5 dump of all file systems. This tape contains all changes made to files since the Level 0 dump, approximately a week's worth of changes. This tape must be saved until the second Friday of the next month, when it will be used again.

g. Steps c-e for the next week, and so on until the end of the month, when the next Level 0 dump is performed.

h. Step f is performed for the third and fourth Fridays of the month using Tapes G and H, respectively, which are saved to be used on the third and fourth Fridays of the next month.

2.8.2.2 Tapes and Equipment. Typical backups for SunOS use either 8mm cassette or 1/4 inch cartridge tape. Tape size depends on the tape device and the tape controller board that is connected to the tape device.

Tapes must be labeled after backups. If a backup strategy similar to the one suggested in the previous subparagraph is planned, indicate on the label "Tape A," "Tape B," etc. Every time a dump is performed, make another tape label containing the backup date, file system backed up, dump level, plus any site-specific information that is created. Tapes should be kept in a safe location, where they will be free of dust and away from magnetic equipment.

2.8.2.2.1 Tape Drives. The three tape controllers for the DSRS systems are listed in Table 2-X. (Also, listed in the table is the device abbreviation as it appears in the **/dev** directory, and the width of the tape it supports.)

Table 2-X. Sun Tape Controllers

Tape Controller	Device Abbrev	Width
Xylogics 472	mt	1/2 inch

Tapemaster	mt	1/2 inch
SCSI	st	1/4 inch

The status feature of the **mt (1)** command will be utilized to determine what type of tape drive will be used. For example:

a. A tape shall be loaded in the drive on which the user will want information.

b. The command **mt -f /dev/*tape* status** will need to be entered where *tape* is either rmt0 or rst0. A second or third drive, when supported, will be rst1 and rst2, respectively. Table 2-XI lists the tapes on a SCSI tape controller.

Table 2-XI. Tapes on a SCSI Tape Controller

Tape	Description	Format	Tracks	Tape Length	Capacity
rst0	Sun-3, Sun-4	QIC-11	9	450 ft	45 Mbytes
rst0	Sun-3, Sun-4	QIC-11	9	600 ft	60 Mbytes
rst8	Sun-3, Sun-4	QIC-24	9	450 ft	45 Mbytes
rst8	Sun-3, Sun-4	QIC-24	9	600 ft	60 Mbytes
rst0	Sun-3, Sun-4	QIC-150	18	600 ft	150 Mbytes
rst0	Sun-3, Sun-4	8mm	N/A	6000 ft	2.3 Gbyte
rst8	Sun-3, Sun-4	8mm	N/A	13000 ft	5 Gbytes

2.8.2.2.2 Determining Available Tape Drives. Each tape device will have a unique device name, such as **st0** or **st1**, when it is properly connected to the workstation. To determine what type devices are available to your system, look at the files **/var/adm/messages** or **/var/adm/messages.***. Each tape device will be listed as shown below when using the following **grep** command:

 # grep st /var/adm/messages

```
May 3 16:11:34 hostname vmunix: st0 at esp1 target 5 lun 0
May 3 16:27:43 hostname vmunix: st0: <Exabyte EXB-8500 8mm Helical Scan>
May 3 16:11:34 hostname vmunix: st2 at esp1 target 4 lun 0
May 3 16:27:43 hostname vmunix: st2: <Archive QIC-150>
```

For systems that have an 8mm tape drive available (defined in the above example as device **st0** <Exabyte EXB-8500 8mm Helical Scan>), the tape drive may be used in high-density (5.0 Gbyte format) mode or low-density (2.3 Gbyte format) mode. Table 2-XII shows the sixteen available tape unit assignments for low-density and high-density modes, eight per mode.

Table 2-XII. Tape Unit Assignments for 8mm Drives

Tape Unit	Low-Density (2.3 Gbyte format)	High-Density (5.0 Gbyte format)
st0	/dev/rst0	/dev/rst8
st1	/dev/rst1	/dev/rst9
st2	/dev/rst2	/dev/rst10
st3	/dev/rst3	/dev/rst11
st4	/dev/rst4	/dev/rst12
st5	/dev/rst5	/dev/rst13
st6	/dev/rst6	/dev/rst14
st7	/dev/rst7	/dev/rst15

2.8.2.3 Using the Dump Command. The dump command is used in the same fashion whether reel-to-reel or cartridge tapes are being backed up. However, the tape configuration and disk device names must be supplied as arguments and different options for the types of tape must be indicated. Refer to Table 2-X and 2-XI for proper device abbreviations and other tape-specific information.

The syntax of the dump command is:

 # /usr/etc/dump *options tape_device_name filesystem_to_dump*

An example of a Level 0 dump command to an 8mm tape is:

 # /usr/etc/dump 0udbsf 54000 126 13000 /dev/nrst0 /dev/rsd0a

The italicized arguments on the command line are described below.

NOTE: If the **f** option is specified, then a value for the argument, *tape_device_name*, needs to be added. This represents the device abbreviation for the tape device on your system. Refer to paragraph 2.8.2.2 for these device abbreviations. When specifying the tape device name, the letter **n** may also be typed directly before the name, as in **/dev/nrst0**, to indicate "no rewind". The advantage of specifying "no rewind" is that copies of more than one file system can be made on the tape during a backup procedure. The disadvantage of specifying "no rewind" is that space may run out during the dump (the tape does not rewind before the dump needs a new tape).

Table 2-XIII. Dump Command Options

Title	Description
options	refers to a series of options that can be applied to the dump command. The options most commonly used are summarized below. Refer to the *Sun System & Network Administration* manual for more detailed information.
0-9	Dump level desired.
a	Creates a dump table-of-contents archive in a specified file, *archiv-file*. **restore** then uses this file to determine whether a file is present on a dump tape and, if it is, on which volume it resides.
b	Blocking factor. The default blocking factor for cartridge tapes (c option specified) is 126. The highest blocking factor available with most tape drives is 126.
c	Dumps to cartridge tape. (The default is reel-to-reel.)
d	Tape density. The default is 54000 bpi for 8mm cassette and 1000 bpi for 1/4 inch tape.
f	Dumps file system to the device indicated at the end of the command line. The default is **/dev/st0**.
s	The size of the tape. The default is 13000 feet for 8mm cassette and 700 feet for 1/4 inch tape.
u	Writes the date the dump was successfully completed to the file **/etc/dumpdates**.
v	Rewinds the medium and checks to verify that the medium and the file system are the same. Can be used only on a quiescent file system.
w	Lists the file systems that need backing up. This information is taken from **/etc/dumpdates** and **/etc/fstab**. When this option is used, all other options are ignored and after reporting, dump immediately exits.
W	Shows all the file systems that appear in **/etc/dumpdates** and highlights those file systems that need backing up.

2.8.2.4 Procedure for Dumping. Before the dump procedure may be performed, there are several steps that need to take place to prepare for the dump.

a. For a Level 0 dump, the system must be in single user mode. A supervisor must login and run shutdown (as described in Section 2.8.13). Then, the system needs to be rebooted and brought up in single-user mode (as described in Section 2.8.15.3).

b. For a Level 0 dump on any file system (during single-user mode), **/usr/etc/fsck** must be entered to run the file system checking program. The **fsck** utility checks the file

system for consistency in the file system structure and makes minor repairs. This action ensures that a full dump is reasonably clean.

c. The tape must be loaded onto the tape device.

d. The dump command needs to be executed for the file systems intended for the backup. Refer to Section 2.8.2.3 for additional information on the dump command, or the *Sun System & Network Administration* manual for specific information.

An example of a Level 0 dump command to an 8mm tape is:

 # /usr/etc/dump 0udbsf 54000 126 13000 /dev/nrst0 /dev/rsd0a

e. The command, **/usr/bin/mt offline**, needs to be entered after all the file systems have been backed up. This command tells the system to rewind the tape unit and take it offline. After rewind, the tape must be removed from the tape device and labeled with the date, file system dumped, and dump level.

f. If a Level 0 dump was performed, the system needs to be brought up in multiuser mode by typing the sequence **CTRL d**.

2.8.3 Cleaning Critical File Systems. The critical file systems which must be routinely monitored and occasionally maintained are the '/' file system, and the **/etc/security/audit** file system. When either of these file systems become more than 90% full, the system can slow down.

The **/tmp** directory is located in the '/' file system and is used by the system and users for storing temporary files. To clean out the **/tmp** directory it is recommended that the system be in single-user mode before deleting all files in the **/tmp** directory.

The **/var/adm/dsrs** directory is generally located in the '/' file system, and is used by the DSRS to store the DSRS log files. To clean the **/var/adm/dsrs** directory it is recommended that the system be backed up with a level 0 dump before deleting the **pc_server.log**, **lib_server.log**, and **dsrs_server.log** files. New DSRS log files will be automatically created when the DSRS executables are executed.

A script file **/dsrscm/bin/cleanup_logs** has been provided to automate the deletion of the DSRS log files. The script file **/dsrscm/bin/cleanup_logs** should be inserted at the bottom of the monthly level 0 dump script file, or executed by root after level 0 dumps are performed.

The **/etc/security/audit** file system contains the **audit_data** file which contains the security audit trail. To clean out the **/etc/security/audit** file system the System Administrator must perform a Level 0 dump of the file system, then boot the system into multi-user mode and log in under the SunOS audit account (which will always be audit) and delete the file **/etc/security/audit/audit_data**. A new empty **audit_data** file is automatically created, when the system is rebooted.

2.8.4 Creating SunOS Accounts.

2.8.4.1 Superuser Account. The SunOS operating system provides the special user name, **root**, for system administrative activities. When a user logs in with this name, they become the most privileged user on the system, the *superuser*. The expressions "superuser" and "root" can be used interchangeably. As superuser, the System Administrator has permission to run critical system administration programs and edit sensitive files (privileges that are denied to regular users). A user can become superuser by either: logging in as **root** in response to the login prompt at the console, or typing **su** from a shell where you are logged in under a regular user name. Some tasks that require superuser privileges include:

a. Editing sensitive files like the kernel configuration file or the password file,

b. Changing permissions for files and directories other than those you personally own, and

c. Running programs that enable you to add new users, groups and equipment.

2.8.4.2 User Accounts. The username and the corresponding password are the most critical security barrier in the SunOS environment. The **passwd** and **passwd.adjunct** administrative databases contain user names, encrypted passwords, and other critical information. On a local host, these databases are mapped to the **/etc/passwd** and **/etc/security/passwd.adjunct** files. The **passwd** file contains information about every system and user account that is able to locally log into that host. The **passwd.adjunct** file will contain the encrypted passwords. User accounts are required for DSRS X/Motif users.

2.8.4.3 The Passwd and Passwd.Adjunct Files. When users log into a Sun host, the **login** program consults the **passwd** database and verifies the user name, and consults the **passwd.adjunct** database to verify the password. If the user name is not in the **passwd** database or the password is not correct for the user name, **login** denies the user access to the host. When a user name and correct password is entered, **login** grants the user access to the host.

Each entry in the **/etc/passwd** file has this syntax:

username:##username:uid:gid:gcos-field:home-dir:login-shell

Each entry in the **/etc/security/passwd.adjunct** file has this syntax:

username::::::

The parameters are briefly described in Table 2-XIV. If the SunOS has been installed with the security option, the encrypted password will not appear here. Instead, the encrypted passwords are placed in the **/etc/passwd.adjunct** file which cannot be read by ordinary user.

Table 2-XIV. Parameters for /etc/passwd File Entries

Parameter	Description
username as the *user name*	User or system account login name, also referred as user id.
##username	Password entry defined during C2 security installation.
uid	Account numerical user ID.
gid	Numerical ID of the group to which the account belongs.
gcos-field	User's real name and other identifying information is printed in the user's mail message heading.
home-dir	Full pathname of the account's home directory.
login-shell	Shell the account accesses upon login.

2.8.4.4 Setting up a SunOS Account. The following procedures show how to modify the **/etc/passwd** and **/etc/security/passwd.adjunct** files which creates a DSRS X/Motif user account. Supervisor/Librarian accounts are similar to the account below with the exception of step k where the login-shell is **/bin/csh**.

a. Basic account requirements for the **passwd** file entry include: the **username** must be unique on the local node, the **gid** must be the same as the group gid defined for the DSRS user group, the **gcos-field** is the full name of the user, and the login shell is **/bin/captive.**

b. The user must login as a "superuser".

c. **/etc/passwd** must be edited using a preferred text editor.

d. An entry for the new user must be created on a separate line.

e. The user's requested login name must be typed,

user:

f. The password field must always be defined by **##username** when C2 security is installed.

user:##user:

g. A user ID is added for a user based on the site's policies. A user ID that already exists in the network domain shall not be used. The user IDs for the root and daemon system accounts use 0 or 1; therefore, 0 or 1 shall not be used for other user IDs.

user:##user:235:

h. A group ID number must be added for the group to which the new user will belong. The UNIX group **staff** is the default. Its group ID is 10.

user:##user:235:10:

i. The user's name must be typed as he or she requested, followed by a colon.

user:##user:235:10:DSRS User:

j. The full pathname of the user's home directory must be typed. This shall be **/home/user_name**, followed by the delimiting colon.

user:##user:235:10:DSRS User:/home/dsrsuser:

k. The user's preferred login shell should be **/bin/captive**. The preferred login shell cannot be the Bourne shell.

user:##user:235:10:DSRS User:/home/dsrsuser:/bin/captive

l. The **/etc/passwd** file must be closed, then a home directory for the new user must be created and the necessary files moved into the user's directory.

 # cd /home
 # mkdir dsrsuser
 # chown *user_id#* dsrsuser
 # cp /dsrscm/config/cshrc /home/dsrsuser/.cshrc
 # cp /dsrscm/config/login /home/dsrsuser/.login
 # cp /dsrscm/config/dsrs /home/dsrsuser/.Xdefaults
 # cp /dsrscm/config/xinitrc /home/dsrsuser/.xinitrc
 # cp /dsrscm/config/mwmrc /home/dsrsuser/.mwmrc
 # chown *user_id#* /home/dsrsuser/.*

The **.cshrc** file will need to be modified to include the DISPLAY environment variable. This will be the IP address of the system for the display of the dsrs X/Motif screens. In addition, the above files will need to be updated with your site-specific configuration.

m. Edit **/etc/security/passwd.adjunct** using a preferred text editor.

n. Create an entry for the new user on a separate line.

o. Type the user's login name, followed by six colons:

user::::::

p. Close the **/etc/security/passwd.adjunct** file.

q. Set the passwd for the newly created SunOS account:

/usr/bin/passwd user

r. Set the password to expire after 60 days:

/usr/bin/passwd -x 60 user

s. An **msql** database must be created which is the same name as the DSRS X/Motif user login name.

 # /usr/local/Minerva/bin/msqladmin create *user*

t. For each **msql** database created with the msqladmin script the following lines must be added to the **/usr/local/Minerva/msql.acl** file.

database = *user*
read = *user*
write = *user*
host = *
access = local, remote

u. Exit from the root shell.

NOTE: The file **/dsrscm/bin/setup.csh** must be modified to point to the correct **UIDPATH** for the SunOS environment. The **UIDPATH** environment variable should be:

./%U:$USR_MOTIF_DIR/bin/sun_os/%U

2.8.5 Creating SunOS Groups.

2.8.5.1 User Groups. Traditional UNIX user groups consist of individuals who use the same set of files and are granted the same set of permissions. This section explains how to set up user groups on the local machine. The system administrator, logged in as **root**, is responsible for modifying the **/etc/group** file.

Two groups are required for the DSRS; these are **dba** and **dsrsadmin**. The **dba** group, created during installation of the ORACLE software, allows users assigned to this group to startup and shutdown the ORACLE databases. The **dsrsadmin** group allows DSRS Librarians and Supervisors access to protected DSRS executables.

2.8.5.2 **The Group File.** In the SunOS environment, the basic form of group protection is the group database. On the local machine, this database takes the form of the **/etc/group** file and the **/etc/security/group.adjunct** file.

Each entry in the **/etc/group** file has this syntax:

groupname:#$groupname:gid:user,user

Terminate each field with a colon, as in the **passwd** file. The **user-list** must be separated with commas and must have no spaces between user names.

The parameters are briefly described in Table 2-XV.

Table 2-XV. Parameters for /etc/group File Entries (Group Access)

Parameter	Description
groupname	Name of the group.
#$groupname	Password entry defined during C2 security installation.
gid	Group's numerical user ID.
user-list	List of users in the group.

2.8.5.3 **Setting Up a SunOS Group.** The following procedures show how to modify the **/etc/group** file that creates SunOS groups. The example shown below is for the group **dsrsadmin**, which includes all users who are librarians and supervisors of the DSRS.

a. Edit **/etc/group** using a preferred text editor.

b. Create an entry for the new group on a separate line.

c. Type the group name:

dsrsadmin:

d. Insert the text **#$groupname** in the next field to indicate that the groups have passwords defined in the **/etc/security/group.adjunct** file:

dsrsadmin:#$dsrsadmin:

e. Add a unique group ID based on the site's policies:

dsrsadmin:#$dsrsadmin:50:

f. Close the **/etc/group** file.

g. Edit **/etc/security/group.adjunct** using a preferred text editor.

h. Create an entry for the new group on a separate line.

i. Type the group name followed by a colon:

dsrsadmin:

j. Insert an asterisk in the next field to indicate that the groups do not have passwords:

dsrsadmin:*

k. Close the **/etc/security/group.adjunct** file.

2.8.5.4 **Assigning Users to Groups.** The last field for each entry in the **/etc/group** file includes all the SunOS account login names of all users who belong to that group. All DSRS Supervisors and Librarians must belong to the **dsrsadmin** group. All SunOS users who need privilege to startup and shutdown ORACLE databases must belong to the **dba** group.

The examples shown below show the DSRS owner account being assigned to the **dsrsadmin** and **dba** groups.

a. Edit **/etc/group** using a preferred text editor.

b. Add the DSRS owner to the last field for each of the groups shown below.

NOTE: When adding users to the group user-list, each user name must be separated by a comma and have no spaces between the entries.

dba:#$dba:30:oracle,root,dsrstest
dsrsadmin:#$dsrsadmin:50:dsrstest

c. Close the **/etc/group** file.

2.8.6 **Debugging DSRS Problems.** If problems are encountered when using the DSRS, the following steps shall be taken to identify the problem and its resolution:

a. The **env** command must be used to verify that the DSRS environment variables have been properly defined. If the variables have not been defined, refer to Section 2.7 for instructions on how to define them.

b. The four database processes that are currently active on the system must be verified. Refer to Section 2.8.7 for instructions on how to determine what processes are available, what processes are not currently active, and how to bring them up with the **dbstart** command.

c. The DSRS database tables must be verified that they have data stored in them. If the tables are empty or corrupt, the database tables need to be restored using the ORACLE import tool described in Section 2.8.9. Refer to Appendix B for a list of the DSRS database tables.

d. If the DSRS variables are properly defined and the ORACLE database is currently active, then SunOS needs to be rebooted. Refer to Section 2.8.13 for steps to shut down the operating system and Section 2.8.15 to start up the operating system.

e. If the DSRS is still not accessible to users, then contact the Customer Assistance Office for further help.

2.8.7 Debugging ORACLE Problems. The following items need to be checked to make sure that the ORACLE database is running properly. If at anytime an ORACLE error message is received, refer to the *ORACLE7 Server Messages and Codes Manual* for actions to resolve the error.

a. It must be verified that the four ORACLE processes (described in Section 2.4.1.6) are currently executing on the system with the **ps -ax** command.

b. If all four processes are executing, then continue to step d. If all four processes are not executing, then execute **dbshut** followed by **dbstart** as described in subsections 2.4.1.3 and 2.4.1.2 respectively, or execute the SQL*DBA tool and enter the command **shutdown abort** and **startup open**.

c. If the database and its four processes do not come up, then refer to the *ORACLE7 Server Messages and Codes Manual* for the action to take to resolve the specific error message received when performing the **dbshut** or **dbstart**.

d. If the database is up and problems still occur, verify that the tablespaces (described in Section 2.4.1) are available and that the DSRS database tables (Appendix A) have data stored in them.

e. If the database tablespaces or the data in the tables is not available, then restore the database information from an ORACLE export file. The steps for restoring database information are detailed in Section 2.8.9.

2.8.8 Debugging SunOS 4.1.3 Problems. Typically, a program shall never crash the host system. However, a user program may crash and "hang" the system (a user process, a user's window, or even the whole system) even though it continues to run. User program crashes, as distinguished from operating system crashes, almost always are caused by an error in the program.

Sometimes the system may appear as hung or dead; that is, it does not respond to anything typed. Before assuming that the program has crashed, check the items below:

a. Type **CTRL Q** in case the screen was frozen by a **CTRL S** keystroke.

b. The **tty** mode may be corrupted. The line feed character may be typed, **CTRL J**, instead of the **RETURN** key, which forces a line feed. If the system responds, **CTRL J**, type **#/usr/ucb/reset CTRL J** to reset the **tty** modes.

c. If the window system is running, make sure the mouse cursor resides in the window where the user is trying to type commands.

d. Type **CTRL **. This shall force a "quit" in the running program and, probably, the writing of a core file.

e. Type **CTRL c** to interrupt the program that may be running.

f. If possible, try logging into the same Central Processing Unit (CPU) from another terminal, or remote login from another system on the network. Type **ps -ax**, and look for the hung process. If hung processes are identified, try to kill the process. To kill the process, the user must be logged in as the same user running the process or have root privileges. Type **kill <pid number>**. If this does not work, try **kill -9 <pid number>**. (A quick way to see if **kill** has worked is to repeat it. If the response is **no such process**, it was killed.)

g. If all of the above steps fail, abort and reboot. Refer to subsection 2.8.15.1 for the reboot command.

h. If it continues to fail, call the Sun Customer Service for help.

2.8.9 Restoring Backups for ORACLE. Backups can be restored for the ORACLE database with the ORACLE Import utility. The Import utility reads data from export files (see Section 2.8.1) into an ORACLE database. The Import utility allows for: import of an entire database, import selected tables for a selected user, import of tables exported by another user, and import of tables from one user to another.

Import can be used in two ways: command-line and interactive method. The command-line method is recommended for restoring the ORACLE database. Only users with DBA privileges can import in full database mode.

ORACLE restores will be performed when necessary. A general example of the command-line import is given below with two parameters specified: (Parameter definitions are detailed in Table 2-XVI).

imp parameter=value parameter=value

Table 2-XVI. Control Parameters for Import Utility (command-line method)

Parameter	Default Value	Description
userid	undefined	The username and password (separated by a slash) of the user performing the import.
full	N	A flag that indicates whether to import the entire file. If N is specified, you are prompted for the names of objects to import.
file	expdat.dmp	The name of the export file to import.
grants	Y	A flag that indicates whether to import grants.
indexes	Y	A flag that indicates whether to import indexes.
ignore	N	A flag that indicates whether to ignore create errors due to the object's existence
show	N	A flag that indicates whether to list only the contents of the export file, and not import the table data, not create any objects, and not modify the database.
rows	Y	A flag that indicates whether to import the rows of table data.
fromuser	undefined	A list of usernames whose objects are imported.
touser	undefined	A list of usernames to whom data is imported.
tables	undefined	A list of table names to import. Use an asterisk (*) to indicate all tables.
commit	N	A flag that indicates whether to commit after each array insert. By default, Import commits after loading each table.

2.8.10 Restoring Backups for SunOS. Restoring files after user deletion or a crash is a significant system administrative activity. If backups (dumps) have occurred on a regular basis, files and file systems can be restored to a reasonable state. If tapes are properly labeled, it is easy to determine which ones to use for the restore process after consulting the **/etc/dumpdates** file. Then, after selection of the proper tape, the following restore procedures are used to recover files.

2.8.10.1 Using the /etc/dumpdates File. The **/etc/dumpdates** file makes a record of each dump performed, provided that the **u** option of dump was specified (as explained in subsection 2.8.2.3).

Each line in **/etc/dumpdates** will give information about the last dump performed at a particular level. The fields in the line include the partition (file system) that was dumped, dump level, and dump date. Refer to the *Sun System & Network Administration Manual* for detailed information about **/etc/dumpdates**.

2.8.10.2 Using restore. The **restore** command copies files from tapes created by the **dump** command into the current working directory. **Restore** is used to reload an entire file system hierarchy from a Level 0 dump and incremental dumps that follow, or to restore one or more single files from any dump tape. The syntax of **restore** is:

 # /usr/etc/restore *key [names...]*

Refer to the *ORACLE7 Server Utility User's Guide* for a complete description and use of all Import parameters.

The *key* refers to one or more of a number of keys available through **restore**. (*keys* differ from options in that they are not preceded by a dash.) These keys are composed of one *function letter* and possibly one or more *function modifiers*. For a complete description of **restore** refer to the *Sun System & Network Administration Manual.*

The most frequently used function letters are listed in Table 2-XVII.

Table 2-XVII. Frequently Used Function Letters

Function Letter	Description
t	Verifies that the files specified on the command line are on the tape. If **restore** finds the files, it displays their names on the screen. If **restore** is run with just the **t** function and no file name, **restore** lists all files on the tape (that is, provides the user with a Table of Contents of the tape).
i	Runs the interactive version of **restore**. The program runs in an interactive environment that lets the user select specific files to be restored.
r	Tells **restore** to do a recursive restore; that is, copy everything on the tape.
x	Restores only the files named on the command line.

The most frequently used modifiers are listed in Table 2-XVIII.

Table 2-XVIII. Frequently Used Function Modifiers

Function Modifiers	Description
v	Displays the name of each file as it is restored-- the verbose option.
f	Indicates that the tape device you want **restore** to read from will be specified on the command line.
a	Takes the dump table-of-contents from the specified archive-file instead of from the dump tape. If the file requested is in the table-of-contents, **restore** prompts the user to mount the tape. If only the contents information is needed (for example, when the **t** option is specified), you do not have to mount the dump tape.
s n	Skip to the n'th file when there are multiple dump files on the same tape.

The ***[names...]*** argument is the name of the file, files, or directories whose files are to be restored.

2.8.11 Procedures for Restoring a File. To verify a file and then restore it:

a. Login as superuser and load the proper dump tape into the tape unit.

b. Go to the directory where the restored files are to be copied.

c. Type the following to verify that the file exists:

 # /usr/etc/restore tf /dev/tape *file_name*

The arguments to /usr/etc/restore are:

t The **t** function letter tells **restore** to verify whether the file, files, or directory given as the *file_name* argument are on the tape.

f The **f** function letter tells **restore** to find the file on the tape device named on the command line.

/dev/tape The name of the tape device used.

file_name The name of the file, files, or directory that you want to restore.

d. Once the files have been found on the tape, restore the file by typing the following:

 # /usr/etc/restore x /dev/tape *file_name*

x The **x** argument tells **restore** to copy the file, files, or directory *file_name.* If *file_name* is a directory, **restore** then recursively extracts the directory.

e. Rewind the tape and take it offline by typing:

 # mt -f /dev/tape offline

2.8.12 Shutdown ORACLE. Refer to subsection 2.4.1.4 for instructions on how to shutdown the ORACLE database.

2.8.13 Shutdown SunOS. The operating system provides several commands for shutting down a system in a non-emergency situation. These commands include:

/usr/etc/shutdown
/usr/etc/halt
/usr/etc/reboot

The user must be superuser to run any of them.

2.8.13.1 The shutdown Command. The **/usr/etc/shutdown** command provides an automated shutdown procedure that notifies users that a system halt is pending. The syntax is:

 # /usr/etc/shutdown [-fhknr] [time [warning-message...]

Use **shutdown** for shutting down a system with multiple users. The user may optionally specify a time to the *time* argument and a message to be broadcast to all current users with the *message* argument. In addition, the various option flags enable a request that will **halt** or **reboot** automatically after shutdown completes. Refer to the *Sun System & Network Administration Manual* for further details on the **shutdown** command.

2.8.13.2 The halt Command. The **/usr/etc/halt** command immediately shuts down the system in an orderly fashion, unless explicitly specified otherwise. The syntax is:

 # /usr/etc/halt [- nqy]

The **halt** command does not have a facility for leaving messages or for specifying a time when it will be executed. The **halt** command writes information to the disks and halts the processor (no warning, no delay). It takes the user out of the operating system and back to the monitor. Use **halt** if the system needs to be brought down quickly and unexpectedly.

Use **shutdown** on a server instead of **halt** unless an immediate system halt is necessary, because users may find this very disturbing.

2.8.13.3 The reboot Command. When the operating system is running and reboot is needed, **shutdown** is normally used on a server or time-shared standalone. However, **/usr/etc/reboot** may

be used on a machine with only one user, or on a server or time-shared standalone currently in single-user mode.

The syntax of reboot is:

 # /usr/etc/reboot [-dnq] [boot_args]

Refer to the *Sun System & Network Administration Manual* for further details on the **reboot** command.

When **reboot** is performed, it executes **sync** to write information to the disk, then initiates a multiuser reboot. During this process, **fsck** performs disk checks. If no problems occur, the system comes up in multiuser mode.

2.8.14 Startup ORACLE. Refer to subsection 2.4.1.2 for instructions on how to startup the ORACLE database.

2.8.15 Startup SunOS. The SunOS can be booted in multiuser and in single-user mode. Multiuser mode is used by all users on a daily basis, single-user mode is used for:

a. Fixing the system to work in multiuser mode. For example, if the **/etc/passwd** file is corrupted, no one can login to the multiuser system. If single-user mode is booted, then the **/etc/passwd** file can be edited from the backup copy and then rebooted to multiuser mode.

b. The file system may be fixed by running **fsck** in single user mode.

c. The **/usr/etc/dump** shall be executed during full system backups.

2.8.15.1 Booting SunOS. The first step in the boot process occurs when the Sun computer is powered up. The CPU in the machine has a PROM containing a program generally known as the *monitor*. The monitor controls the operation of the system before the SunOS kernal takes control.

The monitor runs a quick self-test procedure right after the system is powered on. The automatic boot process is initiated whenever the self-test is completed without errors. The automatic boot can be invoked by doing any of the following:

a. Type the **b** command at the monitor prompt (>).

b. Run the **fastboot** command from the shell (as superuser).

c. Run the **reboot** command from the shell (as superuser).

2.8.15.2 Aborting a Booting Sequence. Occasionally, the booting process may need to be aborted. The automatic boot process only takes place on power-up, or if a **b** command is typed.

The specific abort key sequence depends on the keyboard type. For a Sun-4 keyboard, the sequence is **Stop-A**. For a Sun-3 keyboard, the sequence is **L1-A**. For a standard console terminal keyboard, the **BREAK** key generates an abort.

2.8.15.3 Booting Single-User Mode. It is important to understand how to boot up to run in single-user mode. In this mode, the host only mounts the **/** and **/usr** file system but does not start any daemons nor set the TERM variable.

The host must be booted in single-user mode to fix certain problems and to run certain programs. Also, if boot multiuser mode fails, booting in single-user mode may work.

To reboot **/vmunix** from the default file system and come up in single-user mode, type the **b -s** command at the monitor prompt (>).

The system will send various messages when it checks that expected devices exist, then will stop in single-user mode, giving the pound sign (#) prompt.

When you have completed needed tasks in single-user mode, the system will need to be brought up in multiuser mode by typing the sequence **CTRL D**.

2.8.16 Formatting a Disk.

2.8.16.1 An Overview of format. The SunOS utility **format** enables you to format, label, partition and analyze disks on your system. **Format** provides a friendly, menu-based interface for disk maintenance. For most disk maintenance needs, detailed instructions may be found in the Sun *System & Network Administration* manual.

Sun recommends that each new disk be formatted before it is made available to users. Formatting a disk will erase any information contained on that disk. After a disk has been formatted, it must be partitioned and labeled. Each partition must then be created into a new file system using the **newfs** command. After the new file systems have been created, they may be made available to the users.

Each disk device will have a unique device name, such as **sd0** or **sd4**, when it is properly connected to the workstation. To determine what disk devices are available to your system look at the files **/var/adm/messages** or **/var/adm/messages.***. Each disk device will be listed as shown below when using the following **grep** command:

grep sd /var/adm/messages

```
Apr 27 09:59:27 hostname vmunix: sd0 at esp0 target 3 lun 0
Apr 27 09:59:27 hostname vmunix: sd0: <SUN1.05 cyl 2036 alt 2>
Apr 27 09:59:27 hostname vmunix: sd4 at esp1 target 3 lun 0
Apr 27 09:59:27 hostname vmunix: sd4: <SUN1.05 cyl 2036 alt 2>
```

2.8.16.2 Using the format Command. The following procedure shows an example of how to format a disk. The example below will format the drive which has the device name **sd4**.

a. The user must login as a "superuser".

b. The format command must be executed.

format

c. Format provides a list of each disk available to the system. Choose the proper disk which is to be formatted by typing the corresponding disk number.

AVAILABLE DISK SELECTIONS:
0. sd0 at esp0 slave 24
sd0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
1. sd4 at esp1 slave 24
sd4: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
2. sd5 at esp1 slave 24
sd5: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
Specify disk (enter its number): **1**

d. Format provides a format menu of available options from which you may choose. To format the disk, type the **format** option at the **format>** prompt. The example below is only a brief listing of the available format options.

FORMAT MENU:

disk	- select a disk
partition	- select (define) a partition table
current	- describe the current disk
format	- format and analyze the disk
label	- write label to the disk
defect	- defect list management
quit	

format> **format**

e. Format will respond with the following message. **Yes** (y) must be entered at the Continue? prompt for the format to actually begin.

Ready to format. Formatting cannot be interrupted and takes 45 minutes (estimated). Continue? **yes**

f. Format begins the format and lists the current time. After the format is complete, it begins verifying the media; when the media verification is complete, it will display the format> prompt. To exit the **format** utility, enter **quit**, or refer to Section 2.8.17 to partition the disk.

format> **quit**

g. Exit from the root shell.

2.8.17 Partitioning a Disk.

2.8.17.1 Setting Up or Changing Disk Partitions. Whenever a new disk is set up, it may be necessary or desirable to partition the disk into separate regions, called *partitions*. Partitioning a disk is done after format and, in order to repartition a disk the disk, will need to be relabeled.

The process of partitioning a disk is very simple. Enter **format** and select the disk which will be partitioned. The example below will partition the drive whose device name is **sd4**.

a. The user must login as a "superuser".

b. The format command must be executed.

format

c. Format provides a list of each disk available to the system. Choose the proper disk which is to be partitioned by typing the corresponding disk number.

AVAILABLE DISK SELECTIONS:
0. sd0 at esp0 slave 24
sd0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
1. sd4 at esp1 slave 24
sd4: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
2. sd5 at esp1 slave 24
sd5: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
Specify disk (enter its number): **1**

d. Format provides a format menu of available options from which you may choose. To partition the disk, type **partition** at the **format>** prompt. The example below is only a brief listing of the available format options.

FORMAT MENU:

disk	- select a disk
partition	- select (define) a partition table
current	- describe the current disk
format	- format and analyze the disk
label	- write label to the disk
defect	- defect list management
quit	

format> **partition**

e. Partition provides a partition menu of available options from which the user may choose. To view the current disk partitions, type **print** at the **partition>** prompt. The example below is a listing of the available partition options.

```
PARTITION MENU:
        a       - change 'a' partition
        b       - change 'b' partition
        c       - change 'c' partition
        d       - change 'd' partition
        e       - change 'e' partition
        f       - change 'f' partition
        g       - change 'g' partition
        h       - change 'h' partition
        select  - select a predefined table
        name    - name the current table
        print   - display the current table
        label   - write partition map and label to disk
        quit

        partition> print
```

f. The current partition table will look similar to the example below.

```
Current partition table (original sd4):
partition a - starting cyl   0, # blocks     0 (0/0/0)
partition b - starting cyl   0, # blocks     0 (0/0/0)
partition c - starting cyl   0, # blocks     2052288 (2036/0/0)
partition d - starting cyl   0, # blocks     0 (0/0/0)
partition e - starting cyl   0, # blocks     0 (0/0/0)
partition f - starting cyl   0, # blocks     0 (0/0/0)
partition g - starting cyl   0, # blocks     2052288 (2036/0/0)
partition h - starting cyl   0, # blocks     0 (0/0/0)
```

g. The current sizing has the whole disk being defined as the "g" partition. To create a new partition ("a"), you have to decrease the size of the "g" partition while increasing the size of the "a" partition. First, decide how big you want to make the new partition.

The Sun *System & Network Administration* manual gives the following description on how to define the size of a partition. In addition, here are some facts which might be useful to know: 1 Kilobyte = 1024; 1 Megabyte = 1024 Kilobytes. Sun's blocks are 1/2 (.5) kilobytes each: 2 blocks = 1 Kilobyte; and 2048 blocks = 1 Megabyte.

If the software that you are installing requires 4 Megabytes of disk space, then the partition will have to be large enough to hold that. Now you must determine how many disk blocks are needed to provide the required space. Since SunOS disk blocks

are 512 bytes in size, you need 8192 blocks of space. In addition, partitions must always start at the beginning of a cylinder. So what is the minimum number of cylinders that will give you 8192 blocks? The exact number is dependent on the geometry of the disk. You can determine this number by dividing the number of blocks in the disk, as shown for the "c" partition (the entire disk), by the number of cylinders on the disk. In this example, the Sun drive uses (2052288/2036) or 1008 blocks per cylinder, so the number of needed cylinders is (number of needed blocks/number of blocks per cylinder) or (8192/1008). As this number is slightly higher than 8, we must use 9 cylinders.

h. Now, set the "a" partition to start at the beginning of the disk, put 9 cylinders into the disk, and have the "g" partition take up the rest of the space:

```
partition> a

partition a - starting cyl   0, # blocks        0 (0/0/0)

Enter new starting cyl [0]: 0
Enter new # blocks [0, 0/0/0]: 9/0/0

partition> g

partition g - starting cyl   0, # blocks  2052288 (2036/0/0)

Enter new starting cyl [0]: 9
Enter new # blocks [2052288, 2036/0/0]: 2027/0/0
partition>
```

i. Now all that remains to be done is to label the disk with the new partition information using the **label** command from the partition menu. Format will respond with the following message; **yes** (y) must be entered at the continue? prompt for the labeling to actually begin.

```
partition> label
Ready to label disk, continue? yes
```

j. To exit the partition menu enter **quit**:

```
partition> quit
```

k. To exit the format utility enter **quit**:

```
format> quit
```

l. Exit from the root shell:

To make use of the partitions you just created, you will need to run **newfs** on each partition. Refer to Section 2.8.18 for running **newfs**.

2.8.18 Making a New File System.

2.8.18.1 Using the newfs Command. The SunOS command, **newfs**, creates a new file system. **Newfs** is a "friendly" front-end to the **mkfs** program. On Sun systems, the disk type is determined by reading the disk label for the specified *raw-special-device*.

Raw-special-device is the name of a raw special device residing in **/dev**, including the disk partition, where you want the new file system to be created. If you want to make a file system on **sd0[a-h]**, specify **sd0[a-h]**, **rsd0[a-h]** or **/dev/rsd0[a-h]**; if you only specify **sd0[a-h]**, **newfs** will find the proper device.

The following procedure shows an example of how to create the file systems on the **sd4a** and **sd4g** partitions.

a. The user must login as a "superuser".

b. The newfs command must be executed for the partition sd4a.

```
# /usr/etc/newfs sd4a
```

c. The newfs command must be executed for the partition sd4g.

```
# /usr/etc/newfs sd4g
```

d. Exit from the root shell.

2.8.19 Mounting the File Systems.

2.8.19.1 Making Mount Points. Mount points are locations within a directory tree through which your computer accesses mounted hierarchies. The system can mount these hierarchies locally from a disk or tape, or remotely from another computer on the network. Any directory can serve as a mount point. The mount points for /, **/usr**, and **/home** are provided automatically for you at installation time, together with an extra mount point arbitrarily called **/mnt**.

If you need further mount points, simply create new directories with the **mkdir** command. The example shows the creation of the **/usr3** and **/usr4** mount points which will be used for the **sd4a** and **sd4g** file systems.

```
# mkdir /usr3
# mkdir /usr4
```

2.8.19.2 The /etc/fstab File. The **/etc/fstab** file contains entries for file systems and disk partitions to mount using the mount command, which is normally invoked by the **rc.boot** script at boot time. This file is used by various utilities that mount, unmount, check the consistency of, dump, and restore file systems. It is also used by the system itself when locating the swap partition.

Each entry consists of a line of the form:

filesystem directory type options freq pass

Table 2-XIX. /etc/fstab Entry Options

Option Name	Description
filesystem	is the pathname of a block-special device
directory	s the pathname of the directory on which to mount the filesystem
type	is the filesystem type, which can be one of:
4.2	to mount a block-special device
lo	to loopback-mount a file system
nfs	to mount an exported NFS file system
swap	to indicate a swap partition
ignore	to have the mount command ignore the current entry
rfs	to mount an RFS file system
tmp	filesystem in virtual memory to mount an ISO 9660 Standard or High Sierra Standard CD-ROM file system
options	contains a comma-separated list (no spaces) of mounting options of which can be applied to all types of file systems, and others which only apply to specific types.
The common options are:	
ro	mount the filesystem as read-only
rw	mount the filesystem as read-write
suid	setuid execution allowed
nosuid	setuid execution disallowed
grpid	creates files with BSD semantics for propagation of the group ID
noauto	do not mount this file system automatically
freq	is the interval (in days) between dumps.
pass	is the **fsck** pass in which to check the partition. Filesystems with *pass 0* are not checked.

2.8.19.3 Updating the /etc/fstab File. The **/etc/fstab** file must be modified by the superuser. An example of an **/etc/fstab** file appears below. Included at the end of this file are the two new entries for the disk partitions **sd4a** and **sd4g** created in the Section 2.8.18.

```
/dev/sd0a     /        4.2 rw 1 1
/dev/sd0g     /usr     4.2 rw 1 2
/dev/sd5c     /usr2    4.2 rw 1 2
/dev/fd0      /pcfs    pcfs rw,noauto 0 0
/dev/sd4a     /usr3    4.2 rw 1 2
/dev/sd4g     /usr4    4.2 rw 1 2
```

2.8.19.4 Mounting the File Systems. To mount everything in the current **/etc/fstab** file, the mount command can be used with the **-a** option. The following command will mount all the file systems listed in the **/etc/fstab** file.

 # mount -a

2.8.20 Reconfiguring the Kernel.

2.8.20.1 Why Reconfigure the Kernel. To use multiple ORACLE databases simultaneously, you will need to increase some of the SunOS kernel parameters. A detailed description of the SunOS kernel configuration file is contained in the Sun *System & Network Administration* manual.

2.8.20.2 Modifying the Kernel Configuration Files. This procedure will describe the steps necessary to modify the kernel configuration file.

a. The user must login as a "superuser".

b. To determine the current kernel configuration files that pertain to your model of Sun computer, look at the files **/var/adm/messages.***. Output will be listed as shown below when using the following grep command:

 # grep SunOS /var/adm/messages

```
May  3 19:27:43 hostname vmunix: SunOS Release 4.1.3 (GENERIC) #3: Mon Feb 4
Apr 28 18:57:40 hostname vmunix: SunOS Release 4.1.3 (GENERIC) #3: Mon Feb 4
Apr 19 10:25:23 hostname vmunix: SunOS Release 4.1.3 (GENERIC) #3: Mon Feb 4
```

c. The kernel configuration filename is listed in parentheses when the system is booted. In the example above the kernel configuration filename is GENERIC.

d. Type the following to go to the directory containing the kernel configuration file found in Step b. The third directory specified below will be either **sun4**, **sun4c** or **sun4m** depending on the system architecture.

 # cd /usr/sys/sun[*4,4c,4m*]**/conf**

For example, if the system has a **sun4c** architecture, enter the command below:

 # cd /usr/sys/sun4c/conf

e. Copy the configuration file (in the example above it is GENERIC) to a new file. Call the new file *SYS_NAME*, where *SYS_NAME* represents the name you want to give your configuration file.

 # cp GENERIC *SYS_NAME*

f. Change permissions for *SYS_NAME* as follows:

 # chmod +w *SYS_NAME*

g. Edit ***SYS_NAME*** using your perferred text editor. Append the following lines to the end of the file to allow three concurrent ORACLE databases:

```
#
#       ORACLE V7 parameters
#
options SEMMNI=15
options SEMMSL=25
options SEMMNS=350
```

h. The line below may be updated to increase the maximum number of users to be supported. This line is found near the beginning of the file and the number of users may be increased, if desired.

maxusers 64

i. Close the ***SYS_NAME*** file.

2.8.20.3 Building the Kernel. This section explains how to build the kernel. Building a kernel is easy to do, tailors the kernel to fit your environment, and significantly improves performance. This procedure describes the steps necessary to build the kernel.

a. The user must be a "superuser" and must be located in the directory specified in step d of Section 2.8.20.2.

b. Execute the following command to configure the kernel configuration file modified in step g of Section 2.8.20.2.

 # config SYS_NAME

c. Move to the directory which was created by the previous step.

 # cd ../SYS_NAME

d. Execute the command below to make the kernel.

 # make

e. Install the kernel after the **make** successfully completes by copying the current kernel to a new filename and copying the newly created kernel to the / directory. Execute the commands below to perform these steps.

 # mv /vmunix /vmunix.orig
 # cp vmunix /vmunix

f. Write down the path and filename of the original kernel so that it will be available to you if necessary.

g. The system must be rebooted after installing the new kernel as follows:

 # /usr/etc/fastboot

h. Exit from the root shell.

2.8.21 Configuring NIS. This section explains how to configure NIS on SunOS. Detailed instructions can be found in Chapter 16 of the *Sun System and Network Administration* manual.

To set up a NIS Master Server the **/usr/etc/yp/ypinit** shell script helps establish the master and slave servers. This procedure describes the steps to configure NIS.

a. Boot the system in a single-user mode.

b. Verify that the domain name is set with the command **domainname** defines the domain name which is found in the file **/etc/defaultdomain**. For example, to set the domain name to ims.disa.mil you would enter:

 # domainname ims.disa.mil

c. Type the following to build a fresh set of NIS maps on the master server:

 # cd /var/yp
 # /usr/etc/yp/ypinit -m

d. **ypinit** asks whether you want the procedure to terminate at the first non-fatal error or continue despite non-fatal errors.

If you choose the first option, **ypinit** will exit upon encountering the first problem; you can then fix it and restart **ypinit**. If you prefer to continue, you can try to fix by hand all problems that may occur, then restart **ypinit**.

e. **ypinit** prompts for a list of other hosts to become NIS servers. Enter the name of all other NIS servers.

f. Once **ypinit** has constructed the list of servers, it starts make (1). This program uses instructions contained in the **Makefile** file. If there are problems with the **/var/yp/Makefile** then the steps below can create an NIS Makefile, and the ypinit can be restarted at step c.

(1) The user must be a "superuser" and must be located in the / directory.

(2) Copy the **/dsrscm/utils/nis.tar.Z** file to the / directory.

 # cp /dsrscm/utils/nis.tar.Z /nis.tar.Z

(3) Uncompress the above file with the following command.

 # /usr/ucb/uncompress /nis.tar.Z

(4) Extract the files from the nis.tar file.

 # tar -xvf /nis.tar

g. Type **ypserv** to start providing NIS Services.

 # ypserv

h. Type **ypxfrd** to start the high speed transfer daemon.

 # ypxfrd

2.8.22 <u>Configuring the WWW Server.</u> This section explains how to configure the WWW Server on SunOS.

a. The user must be a “superuser”.

b. Create a **/WWW** directory for the installation of the WWW server.

 # mkdir /WWW

c. Move to the **/WWW** directory.

 # cd /WWW

d. Copy the **/dsrscm/utils/WWW.tar.Z** file to the **/WWW** directory.

 # cp /dsrscm/utils/WWW.tar.Z /WWW/WWW.tar.Z

e. Uncompress the above file with the following command.

 # /usr/ucb/uncompress /WWW/WWW.tar.Z

f. Extract the files from the **WWW.tar** file.

 # tar -xvf /WWW/WWW.tar

g. A README file will be created. This file include installation steps to compile the WWW server tape **make**.

 # make

h. To run the WWW Server, add the following lines to the end of the **/etc/rc.local** file:

```
if [-f /WWW/Daemon/Sun4/httpd ]; then
        /WWW/Daemon/Sun4/httpd &
fi
```

2.8.23 Saving ORACLE Audit Data. This section explains the recommended procedure for saving ORACLE audit data. When ORACLE audit is enabled, ORACLE writes all audit data to the **SYS.AUD$** table and provides predefined views for easy viewing of the table data.

The maximum size of the database audit trail (SYS.AUD$ table) is predetermined during database creation. By default, up to 99 extents, each 10K in size, can be allocated for this table. ORACLE recommends deleting records from the database audit trail to free audit trail space and to facilitiate audit trail management.

The script file **/dsrscm/testdb/export_audit_tables** will export the **SYS.AUD$** table data into an export file and then delete the rows in the **SYS.AUD$** table. This script file should be executed on a monthly basis. It should be executed before a level 0 dump is performed and it requires the DSRS database to be available.

The script file **/dsrscm/testdb/export_audit_tables** will create an export file in the **/var/adm/dsrs** directory with the filename format of **DDMMMYY.aud**. This export file will contain only the **SYS.AUD$** table data.

The DDMMMYY.aud files in the /var/adm/dsrs directory should be deleted if they are greater than a year old to free disk space.

The export_audit_tables script can be executed by any account that belongs to the **dba** and **dsrsadmin** groups. To execute the script to save the ORACLE audit data enter the command:

/dsrscm/testdb/export_audit_tables

2.9 GENERAL SYSTEM ADMINISTRATION-Solaris 2.3. Table 2-XX lists general system administration activities, a description of each, frequency of activity, and a paragraph reference where further information on the activity can be found.

Table 2-XX. General System Administration Activities for Solaris 2.3

ACTIVITY	DESCRIPTION	FREQUENCY	PARAGRAPH REFERENCE
Cleaning File Systems	Steps to clean the critical file systems.	as required	2.9.3
Create Solaris Accounts	Add user accounts to the Solaris.	as required	2.9.4
Create Solaris Groups	Add user accounts to Solaris groups.	as required	2.9.5
Configuring NIS+	Steps to configure NIS	as required	2.9.21
Configuring WWW Server	Steps to configure WWW server	as required	2.9.22
Debug DSRS Problems	Steps to debug the DSRS when executables are not working.	as required	2.9.6
Debug ORACLE Problems	Steps to debug Oracle when there are database problems.	as required	2.9.7
Debug Solaris Problems	Steps to debug the Solaris when there are operating system problems.	as required	2.9.8
Formatting a Disk	Steps to format a disk.	as required	2.9.16
Making a New File System	Steps to create a new file system.	as required	2.9.18
Mounting the File Systems	Steps to automatically mount file systems.	as required	2.9.19
ORACLE Backups	Create database export files.	daily Monday-Friday	2.9.1
Partitioning a Disk	Steps to partition a disk.	as required	2.9.17
Reconfiguring the System	Steps to reconfigure and build a system.	as required	2.9.20
Restore Backups for ORACLE	Recover database information from export file.	as required	2.9.9
Restore Backup for Solaris	Restore Solaris files from backup tapes.	as required	2.9.10
Saving ORACLE Audit Data	Save ORACLE Audit Data	monthly	2.9.23
Shutdown ORACLE	Shutdown the Oracle database.	as required	2.4.1.5
Shutdown Solaris	Shutdown the Solaris.	as required	2.9.13
Startup ORACLE Database	Startup the Oracle database.	as required	2.4.1.3
Startup Solaris	Startup the Solaris.	as required	2.9.15
Solaris Backups	Perform backups on the Solaris environment.	daily Monday-Friday	2.9.2

2.9.1 ORACLE Backups. Backups may be performed for the ORACLE database information with the ORACLE Export utility. The Export utility writes data from the ORACLE database to operating system files in an ORACLE binary format. These files may be read into an ORACLE database to

restore information, using the ORACLE Import utility. Refer to Section 2.9.9 for information on the ORACLE Import utility.

The Export utility can be executed by either command-line or interactive methods. The command-line method is recommended for backing up the ORACLE database because it can be placed in a command file and be run in a background process. Three different modes can be used during the Export utility: Table, User, and Full Database.

ORACLE backups are performed in **Full Database** mode with all table GRANTS and data. An example of the command-line for a Full Database export follows:

exp userid=system/manager full=Y file=dba.dmp grants=Y indexes=Y

2.9.2 Solaris Backups. Backups are one of the most crucial system administration functions. A procedure for regular scheduled backups of all file systems is needed for two major reasons: to ensure file system integrity against possible system crash, and to ensure user files against accidental deletion. If file systems are backed up as scheduled, then the corrupted files can be restored to a reasonably recent state. In Solaris terminology, the words *ufsdump* and dump are often used to mean "back up".

The ufsdump command backs up files on a per-file-system basis. The ufsdump command allows two kinds of dumps: full dumps and incremental dumps. With levels, ranging from 0 to 9, the user may specify whether all files shall be dumped or only those that were altered since the last lower level dump.

A Level 0 ufsdump is a full dump (a backup of the contents of an entire file system). Levels 1 to 9 ufsdumps, perform an incremental dump (a backup of only files that have changed since the last dump of a lower level).

2.9.2.1 Backup Strategies. Table 2-IX illustrates an **EXAMPLE** of a backup plan where users are doing file-intensive work. It assumes a theoretical month that begins on Friday and consists of four, five-day work weeks. A Level 0 ufsdump writes every file in the specified file system to the dump tape. A Level 9 ufsdump writes every file in the specified file system to the dump tape that has changed since the last Level 8 or lower ufsdump. In practice, at least one more Level 9 ufsdump will be necessary, depending upon the number of days in the month. Also, an end-of-month backup may be scheduled on the last Friday or, if applicable, the last weekend of the month.

With this plan, *n* tapes (the number of tapes needed for a full backup of /, /usr, /export, /home) will be used plus eight additional tapes for the incremental dumps. This plan assumes that each incremental dump uses one tape. If large file systems exist, or if there are more than those files systems listed, then more tapes may be needed.

This is how the plan will work:

a. At the end of the month, a full backup (Level 0) of all file systems is created. These tapes need to be saved for a year.

b. On the first Friday of the month, a Level 5 ufsdump of all file systems is created which copies all files changed since the previous lower-level ufsdump. In this case, the Level 0 ufsdump is performed at the end of the month. Tape A is used for this dump, saved for the month, then used for the first Friday of the next month.

c. On the first Monday of the month, Tape B is used to perform a Level 9 ufsdump of all file systems. The ufsdump copies all files changed since the previous lower level dump; in this case, the Level 5 ufsdump that was performed on Friday. Then, Tape B must be stored until the following Monday, when it will be used again.

d. On the first Tuesday of the month, Tape C is used to perform a Level 9 ufsdump of all file systems. Again, the ufsdump copies all files changed since the last lower level ufsdump (Friday's Level 5 dump).

e. Wednesday and Thursday Level 9 ufsdumps must be created on Tapes D and E, respectively.

f. At the end of the week, Tape F is used for a Level 5 ufsdump of all file systems. This tape contains all changes made to files since the Level 0 ufsdump, approximately a week's worth of changes. This tape must be saved until the second Friday of the next month, when it will be used again.

g. Steps c-e for the next week, and so on until the end of the month, when the next Level 0 ufsdump is performed.

h. Step f is performed for the third and fourth Fridays of the month using Tapes G and H, respectively which are saved to be used on the third and fourth Fridays of the next month.

2.9.2.2 Tapes and Equipment. Typical backups for Solaris use either 8mm cassette or 1/4 inch cartridge tape. Tape size depends on the tape device and the tape controller board that is connected to the tape device.

Tapes must be labeled after backups. If a backup strategy similar to the one suggested in the previous subparagraph is planned, indicate on the label "Tape A," "Tape B," etc. Every time a dump is performed, make another tape label containing the backup date, file system backed up, dump level, plus any site-specific information that is created. Tapes should be kept in a safe location, where they will be free of dust and away from magnetic equipment.

2.9.2.2.1 Tape Drives. The two tape controllers for the DSRS systems are listed in Table 2-XXI. (Also, listed in the table is the device abbreviation as it appears in the **/dev** directory, and the width of the tape it supports.)

Table 2-XXI. Sun Tape Controllers

Tape Controller	Device Abbrev	Width
Xylogics 472	rmt	1/2 inch
SCSI	rmt	1/4 inch

The status feature of the **mt (1)** command will be utilized to determine what type of tape drive will be used. For example:

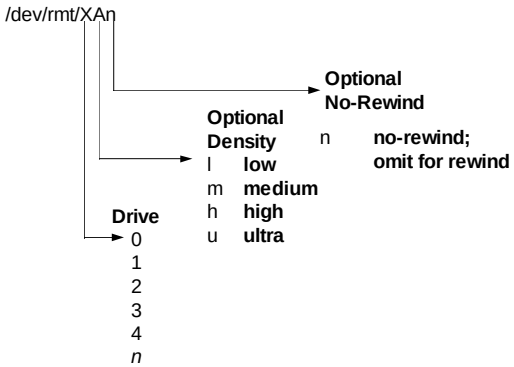
a. A tape shall be loaded in the drive on which the user will want information.

b. The command **mt -f /dev/*tape* status** will need to be entered where *tape* is rmt/0. A second or third drive, when supported, will be rmt/1 and rmt/2, respectively. Table 2-XXII lists the tapes on a SCSI tape controller.

Table 2-XXII. Tapes on a SCSI Tape Controller

Tape	Format	Tracks	Tape Length	Capacity
rmt/01	QIC-11	9	450 ft	45 Mbytes
rmt/01	QIC-11	9	600 ft	60 Mbytes
rmt/0m	QIC-24	9	450 ft	45 Mbytes
rmt/0m	QIC-24	9	600 ft	60 Mbytes
rmt/0h	QIC-150	18	600 ft	150 Mbytes
rmt/0	8mm	N/A	6000 ft	2.3 Gbytes
rmt/0	8mm	N/A	13000 ft	5 Gbytes

2.9.2.2.2 <u>Determining Available Tape Drives.</u> Within the /dev/rmt subdirectory is a single set of tape device files that support different output densities. In general, you specify a tape drive device as shown below.



The next three sections describe drive numbers, the optional density choices, and the optional no-rewind.

Normally, you specify a tape drive by its logical unit number, which is a number from 0 to *n*. If you do not specify a density, the drive writes at its "preferred" density, which is usually the highest density the tape supports.

To specify the first drive, use:

/dev/rmt/0

To specify the second drive, use:

/dev/rmt/1

2.9.2.2.3 Specifying Different Densities for a Tape Drive. The unit and density characters are shown in Table 2-XXIII. For example, to specify a raw magnetic tape device on the first (0) drive with medium density, use:

/dev/rmt/0m

Table 2-XXIII. Unit and Density Characters in Tape Device Names

Device Name	Description
<X>	Tape drive number (digit) from 0 to *n*, regardless of controller type
<A>	Density (character), depending on controller and drive type where <A> may be:
null	Default, preferred (highest) density
l	Low
m	Medium
h	High
u	Ultra

After the command is executed, the tape is automatically rewound unless you specify the no-rewind option as part of the device name. To specify no rewinding, type **n** at the end of the device name.

For example, to specify a raw magnetic tape device on the first (0) drive, no-rewind, with medium density, use:

/dev/rmt/0mn

2.9.2.3 Using the Ufsdump Command. The ufsdump command is used in the same fashion whether reel-to-reel or cartridge tapes are being backed up. However, the tape configuration and disk device names must be supplied as arguments and different options for the types of tape must be indicated. Refer to Table 2-XXII and 2-XXIII for proper device abbreviations and other tape-specific information.

The syntax of the ufsdump command is:

 # /usr/lib/fs/usf/ufsdump *options tape_device_name filesystem_to_dump*

An example of a Level 0 ufsdump command to an 8mm tape is:

 # /usr/lib/fs/usf/ufsdump 0uf /dev/rmt/0n /dev/dsk/c0t3d0s0

The italicized arguments on the command line are described below.

NOTE: If the **f** option is specified, then a value for the argument, *tape_device_name*, needs to be added. This represents the device abbreviation for the tape device on your system. Refer to paragraph 2.9.2.2.2 for these device abbreviations. When specifying the tape device name, the letter **n** may also be typed directly after the name, as in **/dev/rmt/0n**, to indicate "no rewind". The advantage of specifying "no rewind" is that copies of more than one file system can be made on the tape during a backup procedure. The disadvantage of specifying "no rewind" is that space may run out during the ufsdump (the tape does not rewind before the ufsdump needs a new tape).

Table 2-XXIV. Ufsdump Command Options

Title	Description
options	refers to a series of options that can be applied to the ufsdump command. The options most commonly used are summarized below. Refer to the *Sun System & Network Administration* manual for more detailed information.
0-9	Dump level desired.
a	Creates a dump table-of-contents archive in a specified file, *archiv-file*. **ufs restore** then uses this file to determine whether a file is present on a dump tape and, if it is, on which volume it resides.
b	Blocking factor. The default blocking factor for cartridge tapes (c option specified) is 126. The highest blocking factor available with most tape drives is 126.
c	Dumps to cartridge tape. (The default is reel-to-reel.)
d	Tape density. The default is 54000 bpi for 8mm cassette and 1000 bpi for 1/4 inch tape.
f	Dumps file system to the device indicated at the end of the command line. The default is **/dev/rmt/0**.

Title	Description
s	Determine the amount of space that is needed to perform the dump without actually doing it, and display the estimated number of bytes it will take.
u	Writes the date the dump was successfully completed to the file **/etc/dumpdates**.
v	Rewinds the medium and checks to verify that the medium and the file system are the same. Can be used only on a quiescent file system.
w	Lists the file systems that need backing up. This information is taken from **/etc/dumpdates** and **/etc/vfstab**. When this option is used, all other options are ignored and after reporting, dump immediately exits.
W	Shows all the file systems that appear in **/etc/dumpdates** and highlights those file systems that need backing up.

2.9.2.4 Procedure for Dumping. Before the ufsdump procedure may be performed, there are several steps that need to take place to prepare for the dump.

a. For a Level 0 ufsdump, the system must be in single user mode. A supervisor must login and execute shutdown (as described in Section 2.9.13). Then, the system needs to be rebooted and brought up in single-user mode (as described in Section 2.9.15.3).

b. For a Level 0 ufsdump on any file system (during single-user mode), **/usr/etc/fsck** must be entered to execute the file system checking program. The **fsck** utility checks the file system for consistency in the file system structure and makes minor repairs. This action ensures that a full dump is reasonably clean.

c. The tape must be loaded onto the tape device.

d. The ufsdump command needs to be executed for the file systems intended for the backup. Refer to Section 2.9.2.3 for additional information on the dump command, or the *Sun System & Network Administration* manual for specific information.

An example of a Level 0 ufsdump command to an 8mm tape is:

 # /usr/lib/fs/usf/ufsdump 0uf /dev/rmt/0n /dev/dsk/c0t3d0s0

e. The command, **/usr/bin/mt offline**, needs to be entered after all the file systems have been backed up. This command tells the system to rewind the tape unit and take it offline. After rewind, the tape must be removed from the tape device and labeled with the date, file system dumped, and ufsdump level.

f. If a Level 0 ufsdump was performed, the system needs to be brought up in multiuser mode by typing the sequence **CTRL d**.

2.9.3 Cleaning Critical File Systems. The critical file system which must be routinely monitored and occasionally maintained is the '/' file system. When this file system becomes more than 90% full, the system can slow down.

The **/tmp** directory is located in the '/' file system and is used by the system and users for storing temporary files. To clean out the **/tmp** directory it is recommended that the system be in single-user mode before deleting all files in the **/tmp** directory.

The **/var/adm/dsrs** directory is generally located in the '/' file system, and is used by the DSRS to store the DSRS log files. To clean the **/var/adm/dsrs** directory it is recommended that the system be backed up with a level 0 dump before deleting the pc_server.log, lib_server.log, and dsrs_server.log files. New DSRS log files will be automatically created when the DSRS executables are executed.

A script file **/dsrscm/bin/cleanup_logs** has been provided to automate the deletion of the DSRS log files. The script file **/dsrscm/bin/cleanup_logs** should be inserted at the bottom of the monthly level 0 dump script file, or executed by root after level 0 dumps are performed.

2.9.4 Creating Solaris Accounts.

2.9.4.1 Superuser Account. The Solaris operating system provides the special user name, **root,** for system administrative activities. When a user logs in with this name, they become the most privileged user on the system, the *superuser*. The expressions "superuser" and "root" can be used interchangeably. As superuser, the System Administrator has permission to run critical system administration programs and edit sensitive files (privileges that are denied to regular users). A user can become superuser by either: logging in as **root** in response to the login prompt at the console, or typing **su** from a shell where you are logged in under a regular user name. Some tasks that require superuser privileges include:

a. Editing sensitive files like the kernel configuration file,

b. Changing permissions for files and directories other than those you personally own, and

c. Running programs that enable you to add new users, groups and equipment.

2.9.4.2 User Accounts. The username and the corresponding password are the most critical security barrier in the Solaris environment. The **passwd** administrative database contain user names, encrypted passwords, and other critical information. The **passwd** file contains information about every system and user account that is able to locally log into that host. The **/etc/shadow** file will contain the encrypted passwords. User accounts are required for DSRS X/Motif users.

2.9.4.3 The Passwd and Shadow Files. When users log into a Sun host, the **login** program consults the **passwd** database and verifies the user name, and consults the **/etc/shadow** file to verify the password. If the user name is not in the **passwd** database or the password is not correct for the user

name, **login** denies the user access to the host. When a user name and correct password is entered, **login** grants the user access to the host.

Each entry in the passwd database has the fields as described in the table below.

Each entry in the **/etc/shadow** file has this syntax:

username::::::::

The parameters are briefly described in Table 2-XXV. If the Solaris has been installed with the security option, the encrypted password will not appear here. Instead, the encrypted passwords are placed in the **/etc/shadow** file which cannot be read by an ordinary user.

Table 2-XXV. Parameters for passwd Database Entries

Parameter	Description
username as the *user name*	User or system account login name with two to eight characters; also referred as user id.
User ID	Account numerical user ID.
Password Status	You cannot create an account that does not require a password, and you cannot define a password from Database Manager. Use either nispasswd, yppasswd, or passwd to create a password for an account.
Comment (GCOS)	User's real name and other identifying information.
Home Path	Full pathname of the account's home directory.
Shell	Shell the account accesses upon login.
gid	Numerical ID of the group to which the account belongs.
Max Days Valid	Enter the maximum number of days the password will be valid. If you leave the field blank, the password does not expire.
Days Warning	Enter the number of days to begin warning the user before the password expires.
Last Mod Date	The date when the password was last changed is displayed in this field as the number of days since January 1, 1970.
Expiration Date	Enter the absolute date that the user account expires, as the number of days since January 1, 1970. If you leave this field blank, the password does not expire.
Min Change Days	Enter the minimum number of days allowed between password changes.

Parameter	Description
Max Inactive Days	Enter the number of days an account can go unused (no login) before it is locked.

2.9.4.4 Setting up a Solaris Account. The following procedures show how to modify the **passwd** database which creates a user account for either Supervisor/Librarians or DSRS X/Motif users.

a. Basic account requirements for the **passwd** file entry include: the **username** must be unique on the local node, the **gid** must be the same as the group gid defined for the DSRS user group, the **gcos-field** is the full name of the user, and the login shell is **/bin/captive.**

b. The user must login as a "superuser", and have OpenWindows started.

c. Start the Administration tool.

admintool &

d. Click SELECT on the User Account Manager icon.

e. Click SELECT on either the NIS+ naming service or None, and click SELECT on the Load button. The User Account Manager window is displayed.

f. Choose Add User from the Edit menu. The Add User window is displayed.

g. Type the username and user ID in the appropriate text fields.

h. The default password status is to be Normal Password. This allows the password to be set during account creation.

i. Type the rest of the user account information into the text fields. Table 2-XXV describes the contents of each field.

j. The home directory can be created during the account creation by selecting the create home directory block.

k. When you have entered all the information correctly, click SELECT on Add. The information is entered into the Passwd database.

l. To complete the account set-up after editing the Passwd database, you must also move the necessary files into the user's directory.

 # cp /dsrscm/config/cshrc /home/dsrsuser/.cshrc
 # cp /dsrscm/config/login /home/dsrsuser/.login
 # cp /dsrscm/config/dsrs /home/dsrsuser/.Xdefaults

 # cp /dsrscm/config/xinitrc /home/dsrsuser/.xinitrc
 # cp /dsrscm/config/mwmrc /home/dsrsuser/.mwmrc
 # chown *user_id#* /home/dsrsuser/.*

The .cshrc file will need to be modified to include the DISPLAY environment variable. This will be the IP address of the display of the dsrs X/Motif screens. In addition, the above files will need to be updated with your site-specific configuration.

m. Set the passwd for the newly created Solaris account:

(No NIS)	**/usr/bin/passwd**	**user**
(NIS)	**/usr/bin/yppasswd**	**user**
(NIS+)	**/usr/bin/nispasswd**	**user**

n. Set the password to expire after 60 days, with the appropriate password command for your system:

/usr/bin/nispasswd -x 60 user

o. Exit from the root shell.

2.9.5 Creating Solaris Groups.

2.9.5.1 User Groups. Traditional UNIX user groups consist of individuals who use the same set of files and are granted the same set of permissions. This section explains how to set up user groups on the local computer. The system administrator, logged in as **root**, is responsible for modifying the **Group** database file.

Two groups are required for the DSRS; these are **dba**, and **dsrsadm**. The **dba** group, created during installation of the ORACLE software, allows users assigned to this group to startup and shutdown the ORACLE databases. The **dsrsadm** group allows DSRS Librarians and Supervisors access to protected DSRS executables.

2.9.5.2 The Group File. In the Solaris environment, the basic form of group protection is the group database. On the local computer, this database takes the form of the **Group** database file or the **/etc/group** file.

Each entry in the **/etc/group** file has this syntax:

groupname:gid:user,user

Terminate each field with a colon, as in the **passwd** file. The **user-list** must be separated with commas and must have no spaces between user names.

The parameters are briefly described in Table 2-XXVI.

Table 2-XXVI. Parameters for group Database Entries (Group Access)

Parameter	Description
groupname	Name of the group.
gid	Group's numerical user ID.
user-list	List of users in the group.

2.9.5.3 **Setting Up a Solaris Group.** The following procedures show how to modify the **group** database file that creates Solaris groups.

a. The user must login as a "supervisor", and have Open Windows started.

b. Start the Administration tool.

admintool &

c. Click SELECT on the Database Manager icon. The Database Manager window is displayed.

d. Click SELECT on the Group database, then click SELECT on either the NIS+ naming service or None.

e. Click SELECT on Load. The Group Database window is displayed.

f. Choose Add Entry from the Edit Menu. The Group Database: Add Entry window is displayed.

g. Type the group name in the appropriate text field.

h. Type the group ID in the appropriate test field.

i. Type the list of members in the appropriate text field, separated by a comma.

j. Click SELECT on Add. The group is added to the Group database.

k. Exit from the root shell.

2.9.5.4 **Assigning Users to Groups.** The last field for each entry in the **group** database file includes all the Solaris account login names of all users who belong to that group. All DSRS Supervisors and Librarians must belong to the **dsrsadm** group. All Solaris users who need privilege to startup and shutdown ORACLE databases must belong to the **dba** group.

The procedures below show how to modify the **group** database file to add or remove users.

a. The user must login as a "supervisor", and have Open Windows started.

b. Start the Administration tool.

admintool &

c. Click SELECT on the Database Manager icon. The Database Manager window is displayed.

d. Click SELECT on the Group database, then click SELECT on either the NIS+ naming service or None.

e. Click SELECT on the name of the group you want to modify. The Group Database window is displayed.

f. Choose Modify Entry from the Edit Menu. The Group Database: Modify Entry window is displayed, showing the current entries for the group.

g. Add or delete user names from the text field, separated by a comma.

h. Click SELECT on Modify. The entry in the Group database is modified.

i. Exit from the root shell.

2.9.6 Debugging DSRS Problems. If problems are encountered when using the DSRS, the following steps shall be taken to identify the problem and its resolution:

a. The **env** command must be used to verify that the DSRS environment variables have been properly defined. If the variables have not been defined, refer to Section 2.7 for instructions on how to define them.

b. The four database processes that are currently active on the system must be verified. Refer to Section 2.9.7 for instructions on how to determine what processes are available, what processes are not currently active, and how to bring them up with the **dbstart** command.

c. The DSRS database tables must be verified that they have data stored in them. If the tables are empty or corrupt, the database tables need to be restored using the ORACLE import tool described in Section 2.9.9. Refer to Appendix B for a list of the DSRS database tables.

d. If the DSRS variables are properly defined and the ORACLE database is currently active, then Solaris needs to be rebooted. Refer to Section 2.9.13 for steps to shut down the operating system and Section 2.9.15 to start up the operating system.

e. If the DSRS is still not accessible to users, then contact the Customer Assistance Office for further help.

2.9.7 Debugging ORACLE Problems. The following items need to be checked to make sure that the ORACLE database is running properly. If at anytime an ORACLE error message is received, refer to the *ORACLE7 Server Messages and Codes Manual* for actions to resolve the error.

a. It must be verified that the four ORACLE processes (described in Section 2.4.1.6) are currently executing on the system with the **ps -ef** command.

b. If all four processes are executing, then continue to step 4. If all four processes are not executing, then execute **dbshut** followed by **dbstart** as described in subsections 2.4.1.5 and 2.4.1.3 respectively, or execute the SQL*DBA tool and enter the command **shutdown abort** and **startup open**.

c. If the database and its four processes do not come up, then refer to the *ORACLE7 Server Messages and Codes Manual* for the action to take to resolve the specific error message received when performing the **dbshut** or **dbstart**.

d. If the database is up and problems still occur, verify that the tablespaces (described in Section 2.4.1) are available and that the DSRS database tables (Appendix B) have data stored in them.

e. If the database tablespaces or the data in the tables is not available, then restore the database information from an ORACLE export file. The steps for restoring database information are detailed in Section 2.9.9.

2.9.8 Debugging Solaris Problems. Typically, a program shall never crash the host system. However, a user program may crash and "hang" the system (a user process, a user's window, or even the whole system) even though it continues to run. User program crashes, as distinguished from operating system crashes, almost always are caused by an error in the program.

Sometimes the system may appear as hung or dead; that is, it does not respond to anything typed. Before assuming that the program has crashed, check the items below:

a. Type **CTRL Q** in case the screen was frozen by a **CTRL S** keystroke.

b. The **tty** mode may be corrupted. The line feed character may be typed, **CTRL J**, instead of the **RETURN** key, which forces a line feed. If the system responds, **CTRL J**, type **/usr/ucb/reset CTRL J** to reset the **tty** modes.

c. If the window system is running, make sure the mouse cursor resides in the window where the user is trying to type commands.

d. Type **CTRL** \. This shall force a "quit" in the running program and, probably, the writing of a core file.

e. Type **CTRL c** to interrupt the program that may be running.

f. If possible, try logging into the same Central Processing Unit (CPU) from another terminal, or remote login from another system on the network. Type **ps -ef**, and look for the hung process. If hung processes are identified, try to kill the process. To kill the process, the user must be logged in as the same user running the process or have root privileges. Type **kill <pid number>**. If this does not work, try **kill -9 <pid number>**. (A quick way to see if **kill** has worked is to repeat it. If the response is **no such process**, it was killed.)

g. If all of the above steps fail, abort and reboot. Refer to subsection 2.9.15.1 for the reboot command.

h. If it continues to fail, call Sun Customer Service for help.

2.9.9 Restoring Backups for ORACLE. Backups can be restored for the ORACLE database with the ORACLE Import utility. The Import utility reads data from export files (see Section 2.9.1) into an ORACLE database. The Import utility allows for: import of an entire database, import selected tables for a selected user, import of tables exported by another user, and import of tables from one user to another.

Import can be used in two ways: command-line and interactive method. The command-line method is recommended for restoring the ORACLE database. Only users with DBA privileges can import in full database mode.

ORACLE restores will be performed when necessary. A general example of the command-line import is given below with two parameters specified: (Parameter definitions are detailed in Table 2-XVI).

imp parameter=value parameter=value

2.9.10 Restoring Backups for Solaris. Restoring files after user deletion or a crash is a significant system administrative activity. If backups (dumps) have occurred on a regular basis, files and file systems can be restored to a reasonable state. If tapes are properly labeled, it is easy to determine which ones to use for the restore process after consulting the **/etc/dumpdates** file. Then, after selection of the proper tape, the following restore procedures are used to recover files.

2.9.10.1 Using the /etc/dumpdates File. The **/etc/dumpdates** file makes a record of each dump performed, provided that the **u** option of dump was specified (as explained in subsection 2.9.2.3).

Each line in **/etc/dumpdates** will give information about the last dump performed at a particular level. The fields in the line include the partition (file system) that was dumped, dump level, and dump date. Refer to the *Sun System & Network Administration Manual* for detailed information about **/etc/dumpdates**.

2.9.10.2 Using ufsrestore. The **ufsrestore** command copies files from tapes created by the **ufsdump** command into the current working directory. **Ufsrestore** is used to reload an entire file

system hierarchy from a Level 0 dump and incremental dumps that follow, or to restore one or more single files from any dump tape. The syntax of **ufsrestore** is:

 # /usr/lib/fs/usf/ufsrestore *key [names...]*

The *key* refers to one or more of a number of keys available through **ufsrestore**. (*keys* differ from options in that they are not preceded by a dash.) These keys are composed of one *function letter* and possibly one or more *function modifiers*. For a complete description of **ufsrestore** refer to the *Sun System & Network Administration Manual.*

The most frequently used function letters are listed in Table 2-XXVII.

Table 2-XXVII. Frequently Used Function Letters

Function Letter	Description
t	Verifies that the files specified on the command line are on the tape. If **ufsrestore** finds the files, it displays their names on the screen. If **ufsrestore** is run with just the **t** function and no file name, **ufsrestore** lists all files on the tape (that is, provides the user with a Table of Contents of the tape).
i	Runs the interactive version of **ufsrestore**. The program runs in an interactive environment that lets the user select specific files to be restored.
r	Tells **ufsrestore** to do a recursive restore; that is, copy everything on the tape.
x	Restores only the files named on the command line.

The most frequently used modifiers are listed in Table 2-XXVIII.

Table 2-XXVIII. Frequently Used Function Modifiers

Function Modifiers	Description
v	Displays the name of each file as it is restored-- the verbose option.
f	Indicates that the tape device you want **ufsrestore** to read from will be specified on the command line.
a	Takes the dump table-of-contents from the specified archive-file instead of from the dump tape. If the file requested is in the table-of-contents, **ufsrestore** prompts the user to mount the tape. If only the contents information is needed (for example, when the **t** option is specified), you do not have to mount the dump tape.
s n	Skip to the n'th file when there are muliple dump files on the same tape.

The ***[names...]*** argument is the name of the file, files, or directories whose files are to be restored.

2.9.11 Procedures for Restoring a File. To verify a file and then restore it:

a. Login as superuser and load the proper dump tape into the tape unit.

b. Go to the directory where the restored files are to be copied.

c. Type the following to verify that the file exists:

 # /usr/lib/fs/ufs/ufsrestore tf /dev/rmt/tape *file_name*

The arguments to /ufs restore are:

t The **t** function letter tells **ufsrestore** to verify whether the file, files, or directory given as the *file_name* argument are on the tape.

f The **f** function letter tells **ufsrestore** to find the file on the tape device named on the command line.

/dev/rmt/tape The name of the tape device used.

file_name The name of the file, files, or directory that you want to restore.

d. Once the files have been found on the tape, restore the file by typing the following:

 # /usr/sbin/ufsrestore x /dev/tape *file_name*

x The **x** argument tells **ufsrestore** to copy the file, files, or directory *file_name.* If *file_name* is a directory, **ufsrestore** then recursively extracts the directory.

e. Rewind the tape and take it offline by typing:

 # mt -f /dev/rmt/tape offline

2.9.12 Shutdown ORACLE. Refer to subsection 2.4.1.5 for instructions on how to shutdown the ORACLE database.

2.9.13 Shutdown Solaris. The operating system provides several commands for shutting down a system in a non-emergency situation. These commands include:

/usr/sbin/shutdown
/usr/sbin/init
/usr/sbin/halt

/usr/sbin/reboot

The user must be superuser to run any of them.

2.9.13.1 The shutdown Command. The **/usr/sbin/shutdown** command provides an automated shutdown procedure that notifies users that a system halt is pending. The syntax is:

 # /usr/sbin/shutdown [-y] [-g grace-period] [-i init-state]

Use **shutdown** for shutting down a system with multiple users. The user may optionally specify a time to the *time* argument and a message to be broadcast to all current users with the *message* argument. In addition, the various option flags enable a request that will **halt** or **reboot** automatically after shutdown completes. Refer to the *Sun System & Network Administration Manual* for further details on the **shutdown** command.

Table 2-XXIX System Init States

Init State	Function
0	Power-down state
1, S, s	System administrator state (single-user)
2	Multiuser state (resources not exported)
3	Multiuser state (resources exported)
6	Reboot

2.9.13.2 The init Command. To shutdown a single-user system, type **init** and press return. The init command runs scripts that bring down the system cleanly.

2.9.13.3 The halt Command. The **/usr/sbin/halt** command immediately shuts down the system in an orderly fashion, unless explicitly specified otherwise. The syntax is:

 # /usr/sbin/halt [- nqy]

The **halt** command does not have a facility for leaving messages or for specifying a time when it will be executed. The **halt** command writes information to the disks and halts the processor (no warning, no delay). It takes the user out of the operating system and back to the monitor. Use **halt** if the system needs to be brought down quickly and unexpectedly.

Use **shutdown** on a server instead of **halt** unless an immediate system halt is necessary, because users may find this very disturbing.

2.9.13.4 The reboot Command. When the operating system is running and reboot is needed, **shutdown** is normally used on a server or time-shared standalone. However, **/usr/sbin/reboot** may

be used on a computer with only one user, or on a server or time-shared standalone currently in single-user mode.

The syntax of reboot is:

 # /usr/sbin/reboot [-dnq] [boot_args]

Refer to the *Sun System & Network Administration Manual* for further details on the **reboot** command.

When **reboot** is performed, it executes **sync** to write information to the disk, then initiates a multiuser reboot. During this process, **fsck** performs disk checks. If no problems occur, the system comes up in multiuser mode.

2.9.14 Startup ORACLE. Refer to subsection 2.4.1.3 for instructions on how to startup the ORACLE database.

2.9.15 Startup Solaris. The Solaris can be booted in multiuser and in single-user mode. Multiuser mode is used by all users on a daily basis, single-user mode is used for:

a. Fixing the system to work in multiuser mode. For example, if the **/etc/passwd** file is corrupted, no one can login to the multiuser system. If single-user mode is booted, then the **/etc/passwd** file can be edited from the backup copy and then rebooted to multiuser mode.

b. The file system may be fixed by running **fsck** in single user mode.

c. The **/usr/lib/fs/ufs/ufsdump** shall be executed during full system backups.

2.9.15.1 Booting Solaris. The first step in the boot process occurs when the Sun computer is powered up. The CPU in the computer has a programmable read only memory (PROM) chip containing a program generally known as the *monitor*. The monitor controls the operation of the system before the Solaris kernal takes control.

The monitor runs a quick self-test procedure right after the system is powered on. The automatic boot process is initiated whenever the self-test is completed without errors. The automatic boot can be invoked by doing any of the following:

a. Type the **b** command at the monitor prompt (>).

b. Type the **boot** command at the PROM prompt (ok).

c. Run the **fastboot** command from the shell (as superuser).

d. Run the **reboot** command from the shell (as superuser).

2.9.15.2 Aborting a Booting Sequence. Occasionally, the booting process may need to be aborted. Typically, this is done when a boot from a file in a location other than the default disk partition or NFS file system is required. The automatic boot process only takes place on power-up, or if a **b** command is typed.

The specific abort key sequence depends on the keyboard type. For a Sun-4 keyboard, the sequence is **Stop-A**. For a Sun-3 keyboard, the sequence is **L1-A**. For a standard console terminal keyboard, the **BREAK** key generates an abort.

2.9.15.3 Booting Single-User Mode. It is important to understand how to boot up to run in single-user mode. In this mode, the host only mounts the **/** and **/usr** file system but does not start any daemons nor set the TERM variable.

The host must be booted in single-user mode to fix certain problems and to run certain programs. Also, if boot multiuser mode fails, booting in single-user mode may work.

To boot and come up in single-user mode, type the **b -s** command at the monitor prompt (>), or the **boot -s** command at the PROM prompt (ok).

The system will send various messages when it checks that expected devices exist, prompts you for the root password, then will stop in single-user mode, giving the pound sign (#) prompt.

When you have completed needed tasks in single-user mode, the system will need to be brought up in multiuser mode by typing the sequence **CTRL D**.

2.9.15.4 Booting to Configure Device Names. Provides a boot sequence to configure the **/dev** directory when a piece of hardware is added to or removed from the system. Anytime a new piece of hardware is added to a system, it is mandatory to boot the system with the **-r** command before the new hardware will be recognized by the system.

To boot the system after adding or removing hardware, type the **b -r** command at the monitor prompt (>), or the **boot -r** command at the PROM prompt (ok).

2.9.16 Formatting a Disk.

2.9.16.1 An Overview of format. The Solaris utility **format** enables you to format, label, repair, partition and analyze disks on your system. **Format** provides a friendly, menu-based interface for disk maintenance. For most disk maintenance needs, detailed instructions may be found in the Sun *System & Network Administration* manual.

Sun recommends that each new disk be formatted before it is made available to users. Formatting a disk will erase any information contained on that disk. After a disk has been formatted, it must be partitioned and labeled. Each partition must then be created into a new file system using the **newfs** command. After the new file systems have been created, they may be made available to the users.

Each disk device will have a unique device name based on logical (not physical) device names, such as **c0t3d0** or **c0t1d0**, when it is properly connected to the workstation. Each disk device has its own subdirectory in **/dev/dsk**.

To specify a slice (partition) on a disk, use a device name with these conventions:

/dev/dsk/cWtXdYsZ.

Table 2-XXX. Naming Convention for Disks

cWtXdYsZ	Device Naming Conventions
cW	W specifies the slice (or partition) number (0 to 7). If you have only one controller on your system, W is always 0.
tX	X is the target address set by the switch on the back of the unit.
dY	Y is the number of the drive attached to the target.
sZ	Z is the slice (partition) number, a number from 0 to 7. To specify the entire disk, use slice 2. 2 corresponds to partition C under SunOS 4.1.3 disk partitions.

2.9.16.2 **Using the format Command.** The following procedure shows an example of how to format a disk. The example below will format the drive which has the device name **c0t1d0**.

a. The user must login as a "superuser".

b. The format command must be executed.

format

c. Format provides a list of each disk available to the system. Choose the proper disk which is to be formatted by typing the corresponding disk number.

AVAILABLE DISK SELECTIONS:
0. c0tld0 at esp0 slave 24
c0tld0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>

1. c0t2d0 at esp1 slave 24
   c0t2d0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
2. c0t3d0 at esp1 slave 24
   c0t3d0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>

Specify disk (enter its number): **1**

d. Format provides a format menu of available options from which you may choose. To format the disk, type the **format** option at the **format>** prompt. The example below is only a brief listing of the available format options.

FORMAT MENU:

disk	- select a disk
type	- select (define) a disk type
partition	- select (define) a partition table
current	- describe the current disk
format	- format and analyze the disk
repair	- repair a defective sector
label	- write label to the disk
analyze	- surface analysis
defect	- defect list management
backup	- search for backup labels
verify	- read and display labels
save	- save new disk/partition definitions
inquiry	- show vendor, product and revision
quit	

format> **format**

e. Format will respond with the following message. **Yes** (y) must be entered at the Continue? prompt for the format to actually begin.

Ready to format. Formatting cannot be interrupted and takes 45 minutes (estimated). Continue? **yes**

f. Format begins the format and lists the current time. After the format is complete, it begins verifying the media; when the media verification is complete, it will display the format> prompt. To exit the **format** utility, enter **quit**, or refer to Section 2.9.17 to partition the disk.

format> **quit**

g. Exit from the root shell.

2.9.17 Partitioning a Disk.

2.9.17.1 Setting Up or Changing Disk Partitions. Whenever a new disk is set up, it may be necessary or desirable to partition the disk into separate regions, called *partitions*. Partitioning a disk is done after format and, in order to repartition a disk the disk, will need to be relabeled.

The process of partitioning a disk is very simple. Enter **format** and select the disk which will be partitioned. The example below will partition the drive whose device name is **sd4**.

a. The user must login as a "superuser".

b. The format command must be executed.

format

c. Format provides a list of each disk available to the system. Choose the proper disk which is to be partitioned by typing the corresponding disk number.

AVAILABLE DISK SELECTIONS:
0. c0t1d0 at esp0 slave 24
c0t1d0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
1. c0t2d0 at esp1 slave 24
c0t2d0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
2. c0t3d0 at esp1 slave 24
c0t3d0: <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
Specify disk (enter its number): **1**

d. Format provides a format menu of available options from which you may choose. To partition the disk, type **partition** at the **format>** prompt. The example below is only a brief listing of the available format options.

FORMAT MENU:

disk	- select a disk
type	- select (define) a disk type
partition	- select (define) a partition table
current	- describe the current disk
format	- format and analyze the disk
repair	- repair a defective sector
label	- write label to the disk
analyze	- surface analysis
defect	- defect list management
backup	- search for backup labels
verify	- read and display labels
save	- save new disk/partition definitions
inquiry	- show vendor, product and revision
quit	

format> **partition**

e. Partition provides a partition menu of available options from which the user may choose. To view the current disk partitions, type **print** at the **partition>** prompt. The example below is a listing of the available partition options.

PARTITION MENU:

0	- change '0' partition
1	- change '1' partition
2	- change '2' partition
3	- change '3' partition
4	- change '4' partition
5	- change '5' partition
6	- change '6' partition
7	- change '7' partition
select	- select a predefined table
modify	- modify a predefined partition table
name	- name the current table
print	- display the current table
label	- write partition map and label to disk
quit	

partition> **print**

f. The current partition table will look similar to the example below.

Current partition table (original sd4):

Table 2-XXXI. Example Partition Table

Part	Tag	Flag	Cylinders	Size	Blocks
0	unassigned	wm	0	0	(0/0/0)
1	unassigned	wm	0	0	(0/0/0)
2	unassigned	wm	0 - 2101	1.25G	(165/0/0)
3	unassigned	wm	0	0	(0/0/0)
4	unassigned	wm	0	0	(0/0/0)
5	unassigned	wm	0	0	(0/0/0)
6	unassigned	wm	0	0	(0/0/0)
7	unassigned	wm	1 - 2101	1.25G	(165/0/0)

g. The current sizing has the whole disk being defined as the "7" partition. To create a new partition ("0"), you have to decrease the size of the "7" partition while increasing the size of the "0" partition. First, decide how big you want to make the new partition.

The Sun *System & Network Administration* manual gives the following description on how to define the size of a partition. In addition, here are some facts which might be useful to know: 1 Kilobyte = 1024; 1 Megabyte = 1024 Kilobytes. Sun's blocks are 1/2 (.5) kilobytes each: 2 blocks = 1 Kilobyte; and 2048 blocks = 1 Megabyte.

If the software that you are installing requires 4 Megabytes of disk space, then the partition will have to be large enough to hold that. Now you must determine how many disk blocks are needed to provide the required space. Since SunOS disk blocks are 512 bytes in size, you need 8192 blocks of space. In addition, partitions must always start at the beginning of a cylinder. So what is the minimum number of cylinders that will give you 8192 blocks? The exact number is dependent on the geometry of the disk. You can determine this number by dividing the number of blocks in the disk, as shown for the "2" partition (the entire disk), by the number of cylinders on the disk. In this example, the Sun drive uses (2052288/2036) or 1008 blocks per cylinder, so the number of needed cylinders is (number of needed blocks/number of blocks per cylinder) or (8192/1008). As this number is slightly higher than 8, we must use 9 cylinders.

h. Now, set the "0" partition to start at the beginning of the disk, put 9 cylinders into the disk, and have the "7" partition take up the rest of the space:

```
partition> 0

partition 0 - starting cyl   0, # blocks        0 (0/0/0)

Enter new starting cyl [0]: 0
Enter new # blocks [0, 0/0/0]: 9/0/0

partition> 7

partition 7 - starting cyl   0, # blocks  2052288 (2036/0/0)

Enter new starting cyl [0]: 9
Enter new # blocks [2052288, 2036/0/0]: 2027/0/0
partition>
```

i. Now all that remains to be done is to label the disk with the new partition information using the **label** command from the partition menu. Format will respond with the following message; **yes** (y) must be entered at the continue? prompt for the labeling to actually begin.

partition> **label**
Ready to label disk, continue? **yes**

j. To exit the partition menu enter **quit**:

partition> **quit**

k. To exit the format utility enter **quit**:

format> **quit**

l. Exit from the root shell:

To make use of the partitions you just created, you will need to run **newfs** on each partition. Refer to Section 2.9.18 for running **newfs**.

2.9.18 Making a New File System.

2.9.18.1 Using the newfs Command. The command, **newfs**, creates a new file system. **Newfs** is a "friendly" front-end to the **mkfs** program. On Sun systems, the disk type is determined by reading the disk label for the specified *raw-special-device*.

Raw-special-device is the name of a raw special device residing in **/dev/rdsk**, including the disk partition, where you want the new file system to be created. If you want to make a file system on **c0t0d0s[0-7]**, specify **c0t0d0s[0-7]**, or **/dev/rdsk/c0t0d0s[0-7]**; if you only specify **c0t0d0s[0-7]**, **newfs** will find the proper device.

The following procedure shows an example of how to create the file systems on the **c0t0d0s0** and **c0t0d0s7** partitions.

a. The user must login as a "superuser".

b. The newfs command must be executed for the partition c0t0d0s0.

/usr/sbin/newfs c0t0d0s0

c. The newfs command must be executed for the partition c0t0d0s7.

/usr/sbin/newfs c0t0d0s7

d. Exit from the root shell.

2.9.19 Mounting the File Systems.

2.9.19.1 Making Mount Points. Mount points are locations within a directory tree through which your computer accesses mounted hierarchies. The system can mount these hierarchies locally from a disk or tape, or remotely from another computer on the network. Any directory can serve as a mount point. The mount points for **/**, **/usr**, and **/home** are provided automatically for you at installation time, together with an extra mount point arbitrarily called **/mnt**.

If you need further mount points, simply create new directories with the **mkdir** command. The example shows the creation of the **/usr3** and **/usr4** mount points which will be used for the **c0t0d0s0** and **c0t0d0s7** file systems.

```
# mkdir /usr3
# mkdir /usr4
```

2.9.19.2 The /etc/vfstab File. The **/etc/vfstab** file contains entries for file systems and disk partitions to mount using the mount command, which is normally invoked by the **rc.boot** script at boot time. This file is used by various utilities that mount, unmount, check the consistency of, dump, and restore file systems. It is also used by the system itself when locating the swap partition.

Each entry consists of a line of the form:

filesystem directory type options freq pass

Table 2-XXXII. /etc/vfstab Entry Options

Option Name	Description
device to mount	is the pathname of a block-special device
device to fsck	is the pathname of the block-special device to fsck
mount point	is the pathname of the directory on which to mount the filesystem
FS type	is the filesystem type, which can be one of:
4.2	to mount a block-special device
lo	to loopback-mount a file system
nfs	to mount an exported NFS file system
swap	to indicate a swap partition
ignore	to have the mount command ignore the current entry
rfs	to mount an RFS file system
tmp	filesystem in virtual memory to mount an ISO 9660 Standard or High Sierra Standard CD-ROM file system

fsck pass	numer of fsck passes
mount at boot	yes or no to mount at boot
options	contains a comma-separated list (no spaces) of mounting options of which can be applied to all types of file systems, and others which only apply to specific types.

The common options are:	
ro	mount the filesystem as read-only
rw	mount the filesystem as read-write
suid	setuid execution allowed
nosuid	setuid execution disallowed
grpid	creates files with BSD semantics for propagation of the group ID
noauto	do not mount this file system automatically
freq	is the interval (in days) between dumps.
pass	is the **fsck** pass in which to check the partition. Filesystems with *pass 0* are not checked.

2.9.19.3 **Updating the /etc/vfstab File.** The **/etc/vfstab** file must be modified by the superuser. An example of an **/etc/vfstab** file appears below. Included at the end of this file are the two new entries for the disk partitions **c0t0d0s0** and **c0t0d0s7** created in the Section 2.9.18.

```
/dev/dsk/c0t1d0s0   /dev/dsk/c0t1d0s0   /      ufs   1   no
/dev/dsk/c0t1d0s6   /dev/dsk/c0t1d0s6   /usr   ufs   1   no
/dev/dsk/c0t2d0s2   /dev/dsk/c0t2d0s2   /usr2  ufs   2   yes
/dev/dsk/c0t0d0s0   /dev/dsk/c0t0d0s0   /usr3  ufs   2   yes
/dev/dsk/c0t0d0s7   /dev/dsk/c0t0d0s7   /usr4  ufs   2   yes
```

2.9.19.4 **Mounting the File Systems.** To mount everything in the current **/etc/vfstab** file, the mount command can be used with the **-a** option. The following command will mount all the file systems listed in the **/etc/vfstab** file.

```
# /usr/sbin/mountall
```

2.9.20 **Reconfiguring the System.**

2.9.20.1 **Why Reconfigure the System.** To use multiple ORACLE databases simultaneously, you will need to increase some of the Solaris system parameters.

2.9.20.2 **Modifying the System Configuration Files.** This procedure will describe the steps necessary to modify the system configuration file.

a. The user must login as a "superuser".

b. Type the following to go to the directory containing the kernel configuration file.

 # cd /etc

c. Edit System using your perferred text editor. Append the following lines to the end of the file to allow three concurrent ORACLE databases:

```
#
#      ORACLE V7 parameters
#
set shmsys:shminfo_shmmax=8388608
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
set semsys:seminfo_semmns=250
set semsys:seminfo_semmni=70
set semsys:seminfo_semmsl=20
```

d. The line below may be updated to increase the maximum number of users to be supported. This line is found near the beginning of the file and the number of users may be increased, if desired.

set maxusers = 64

e. Close the **/etc/system** file.

f. The system must be rebooted after installing the new system file as follows:

 # /usr/ucb/fastboot

g. Exit from the root shell.

2.9.21 **Configuring NIS+.** This section explains how to configure the Network Information Service Plus (NIS+) on Solaris 2.3. Detailed instructions can be found in the *SunOS 5.3 Administering NIS+ and DNS* manual for Solaris 2.3.

2.9.21.1 **Creating a Root Master Server.** Setting up the root master server is the first step towards establishing an NIS+ domain. The examples below uses ***wiz.com*** as the name of the root domain. The commands below must be executed from the root account.

a. Set the Supervisor's **PATH** variable to include **/usr/lib/nis**.

 # setenv PATH $PATH:/usr/lib/nis

b. Type the following as supervisor (root) to set up a root master server. The **-r** option indicates that a root master server should be set up. The **-d** option specifies the NIS+ domain name.

 # nisserver -r -d *wiz.com*

Type **y** if the information shown on the screen is correct.

Type **y** to continue the NIS+ setup.

Type your computer's root password at the prompt, then press Return.

2.9.21.2 **Populating NIS+ Tables**. This section shows you how to populate the root master server's tables with data from files. It is recommended that you create a directory and make copies of the /etc files and use the copies to populate the tables. This example uses **/nis+files** as the working directory.

a. The files listed below, if they exist, will be copied into the **/nis+files** directory.

 # cp /etc/auto_master /nis+files/auto_master
 # cp /etc/auto_home /nis+files/auto_home
 # cp /etc/aliases /nis+files/aliases
 # cp /etc/ethers /nis+files/ethers
 # cp /etc/netgroup /nis+files/netgroup
 # cp /etc/group /nis+files/group
 # cp /etc/hosts /nis+files/hosts
 # cp /etc/networks /nis+files/networks
 # cp /etc/passwd /nis+files/password
 # cp /etc/protocols /nis+files/protocols
 # cp /etc/services /nis+files/services
 # cp /etc/rpc /nis+files/rpc
 # cp /etc/netmasks /nis+files/netmasks
 # cp /etc/bootparms /nis+files/bootparams

b. Type the following to populate the tables from the files. The **-F** option indicates that the tables will take their data from files. The **-p** option specifies the directory search path for the source files. The **-d** option specifies the NIS+ domain name.

 # nispopulate -F -p /nis+files -d *wiz.com*

2.9.22 **Configuring the WWW Server** This section explains how to configure the WWW Server on Solaris.

1. The user must be a “superuser”.

2. Create a **/WWW** directory for the installation of the WWW server.

   # mkdir /WWW

3. Move to the **/WWW** directory.

   # cd /WWW

4. Copy the **/dsrscm/utils/WWW.tar.Z** file to the **/WWW** directory.

   # cp /dsrscm/utils/WWW.tar.Z /WWW/WWW.tar.Z

5. Uncompress the above file with the following command.

   # /usr/ucb/uncompress /WWW/WWW.tar.Z

6. Extract the files from the **WWW.tar** file.

   # tar -xvf /WWW/WWW.tar

7. A README file will be created. This file include installation steps to compile the WWW server tape **make**.

   # make

8. To execute the WWW Server, add the following lines to the end of the **/etc/rc.local** file:

   ```
   if [-f /WWW/Daemon/Sun4/httpd ]; then
        /WWW/Daemon/Sun4/httpd &
   fi
   ```

2.9.23 **Saving ORACLE Audit Data**. This section explains the recommended procedure for saving ORACLE audit data. When ORACLE audit is enabled, ORACLE writes all audit data to the **SYS.AUD$** table and provides predefined views for easy viewing of table data.

The maximum size of the database audit trail (SYS.AUD$ table) is predetermined during database creation. By default, up to 99 extents, each 10K in size, can be allocated for this table. ORACLE recommends deleting records from the database audit trail to free audit trail space and to facilitate audit trail management.

The script file **/dsrscm/testdb/export_audit_tables** will export the **SYS.AUD$** table data into an export file and then delete the rows in the **SYS.AUD$** table. This script file should be executed on

a monthly basis. It should be executed before a level 0 dump is performed and it requires the DSRS database to be available.

The script file **/dsrscm/testdb/export_audit_tables** will create an export file in the **/var/adm/dsrs** directory with the filename format of **DDMMMYY.aud**. This export file will contain only the **SYS.AUD$** table data.

The DDMMMYY.aud files in the /var/adm/dsrs directory should be deleted if they are greater than a year old to free disk space.

The export_audit_tables script can be executed by any account that belongs to the **dba** and **dsrsadm** groups. To execute the script to save the ORACLE audit data, enter the command:

/dsrscm/testdb/export_audit_tables